University of Alberta
Printing Department

THE UNIVERSITY OF ALBERTA

DEADBEAT RESPONSE BY DIGITAL CONTROLLERS

by

ROBERT CLYDE MARTELL

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

EDMONTON, ALBERTA

FEBRUARY, 1967

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled Deadbeat Response by Digital Controllers submitted by R. Clyde Martell in partial fulfilment of the requirements for the degree of Master of Science.

ABSTRACT

This thesis uses the state transition technique to obtain the z-transforms of digital controllers which, when inserted into the particular system under study, provide an output response that is deadbeat. Various methods of graphically representing the system including the digital controller, writing the state transition equations from the flow graphs and solving the equations are described.

A digital computer program was written to solve the state equations for third- and fourth-order systems. The program was used to obtain the transforms of digital controllers for a number of systems and the results confirmed on the PACE 231-R analog computer.

A method, proposed by Y. J. Kingma and programmed in FORTRAN II by D. H. Kelly, of solving for the roots of a polynomial using Routh's stability criterion was rewritten into FORTRAN IV and double precisioned to provide results with greater accuracy. The program will solve ill-conditioned polynomials and will provide results as accurate as those given by the SHARE Library program C2-3332 based on Newton and Muller approximations.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation for the suggestions and cooperation offered by the staff members and postgraduate students. This project was done under the supervision of Y. J. Kingma, to whom the author wishes to make special acknowledgement.

The author owes a debt of gratitude to his parents for their patience and best wishes.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In the synthesis and design of feedback control systems, the output response must satisfy some predetermined performance. Some of the criteria used in the design of both sampled-data and all-continuous systems are maximum overshoot, the time for the error to reach its first zero, the time to reach maximum overshoot, settling time and the frequency of oscillation of the transient. If the design requirements are such that a rigid performance specification must be met, the usual methods of compensation become cumbersome, time-consuming and depend on the experience of the designer. This is especially true when working with sampled-data systems.

The compensation of sampled-data systems to obtain the desired performance can be accomplished by insertion of continuous-data networks in the forward or feedback paths of the system. Since the network is connected to the controlled process, the z-transform of the product of the two transfer functions is a single function. Analytically, this makes it difficult to determine the effect of various networks on the overall performance. As an alternative, digital compensation can be implemented through the use of digital controllers. Digital controllers are in general more versatile than continuous-data networks and better system performance can thus be achieved. They can be used in continuous-data systems as well as sampled-data systems.

Digital controllers may be synthesized using passive networks preceded and followed by sampling switches, by

digital computer techniques or by employing both analog and digital components. Its main function is to process a sequence of numbers equally spaced in time into a command signal which is then applied to the control process.

Using this method of compensation, quite stringent performance specifications can be met, such as minimal and deadbeat response. Minimal response requires the system to have:

1. Zero steady-state error at the sampling instants for a specified input signal.
2. A rise time equal to a minimum number of sampling periods.
3. A finite settling time measured at the sampling instants.

The design requirements for deadbeat response are more rigid than for minimal response. They are as follows:

1. The system must have zero steady-state error after a minimum number of sampling instants with no intersampling ripple for a specified input signal.
2. The transient response should be as fast as possible, with a finite settling time measured at the sampling instants.

Deadbeat response is a form of minimal response with the further restriction of no intersampling ripple. The ensuing chapters will demonstrate the synthesis procedures for obtaining deadbeat response using digital controllers.

To design a digital controller for deadbeat response,

the state transition approach will be used as it provides a simple and systematic method for continuous-data as well as sampled-data systems. The procedure will be developed using the transition flow graphs and equations, and a digital program will be described for solving the state equations. The method is easy to use and requires little design experience. Although the method can be applied to systems having real, finite poles with assurance of obtaining the desired results, a judicious choice of sampling rate is required to obtain a stable output when the system contains complex conjugate roots.

Since a knowledge of the roots of the plant polynomial is sometimes required, a method of determining them using Routh's stability criterion is described. Knowing the roots, the designer can then choose an appropriate sampling rate to obtain the desired deadbeat response.

CHAPTER II

Z-TRANSFORMS OF DIGITAL CONTROLLERS USING THE STATE TRANSITION TECHNIQUE

There are two major approaches to system analysis and synthesis: 1) the use of block diagrams and transfer functions, and 2) the state transition or state variable approach using signal flow graphs. The state transition approach will be used in this thesis since the system can be represented by a set of first-order differential or difference equations describing the state variables of the system. The continuous elements are described by differential equations, the discrete elements by difference equations, and the sample-and-hold elements by discrete state transition equations.

Once mathematical expressions have been obtained for all the system elements, the state transition flow graph can be drawn. With the aid of the flow graph, the state transition equations can readily be written. Solving these equations manually is time consuming, but since they are linear difference equations, they can be solved using a digital computer.

2.1 BLOCK DIAGRAM REPRESENTATION OF CONTROL SYSTEM

If a digital controller is used to compensate a sampled-data system, its block diagram representation consists of the Laplace transform of the controller followed by a sampling switch synchronized with the other system switches. The block diagram is drawn as shown in Figure 2-1.

If the system to be compensated is continuous, the block diagram representation of the digital controller

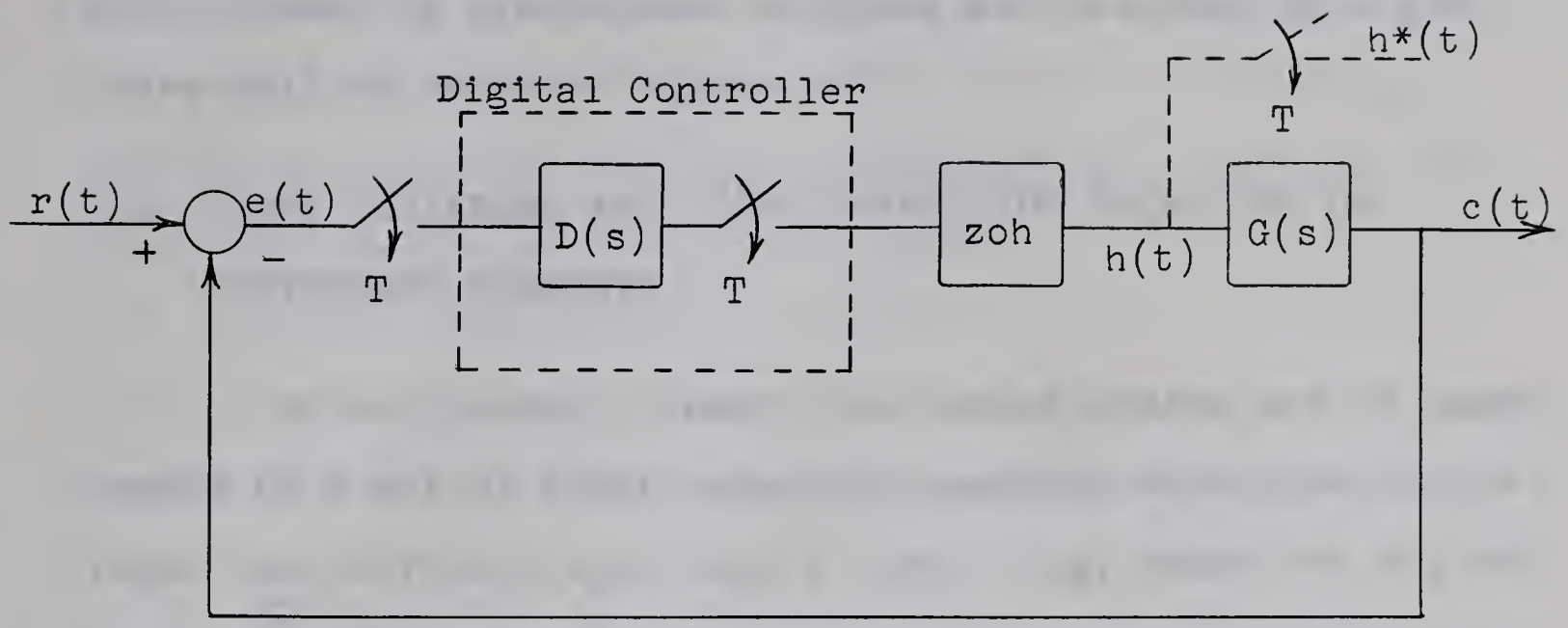


FIGURE 2-1. SAMPLED-DATA SYSTEM

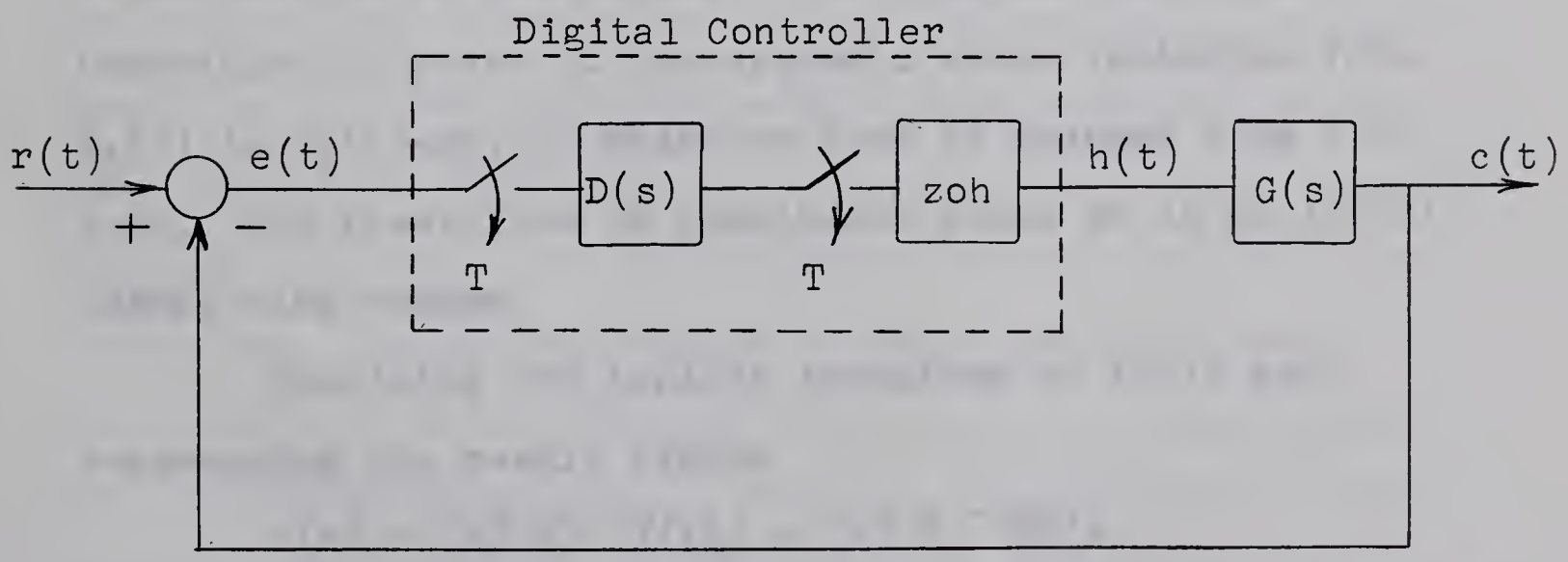
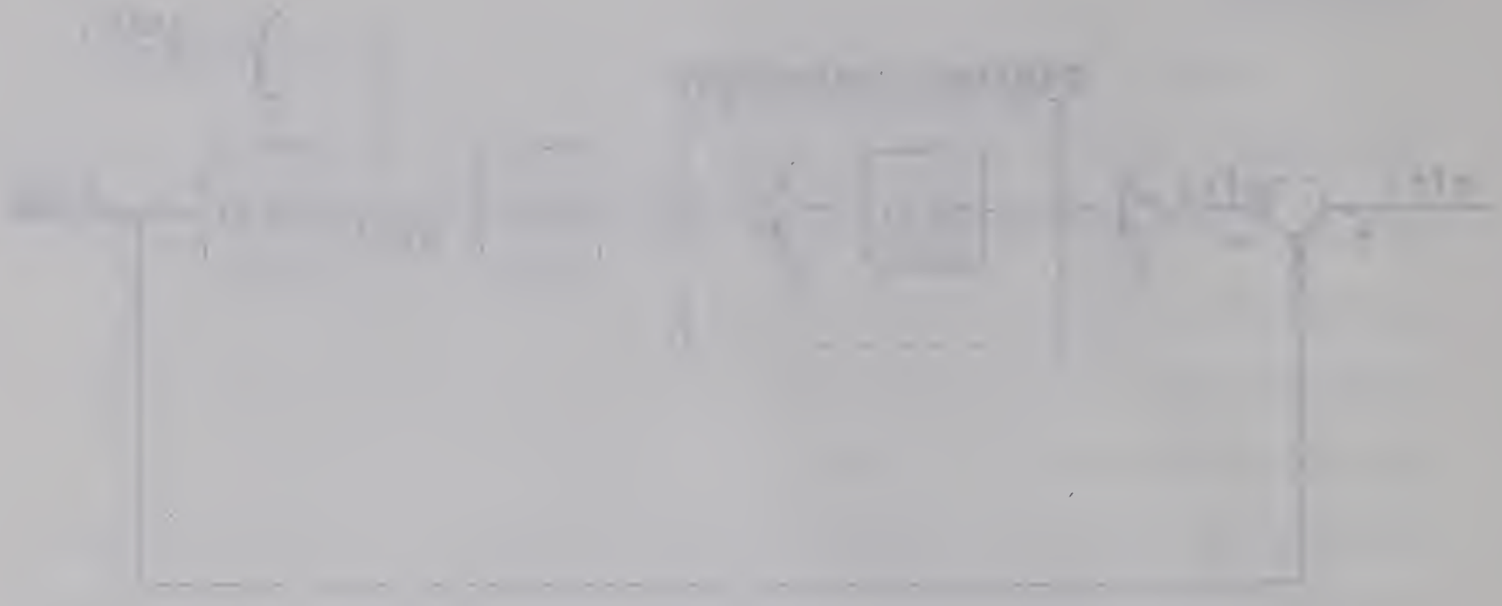
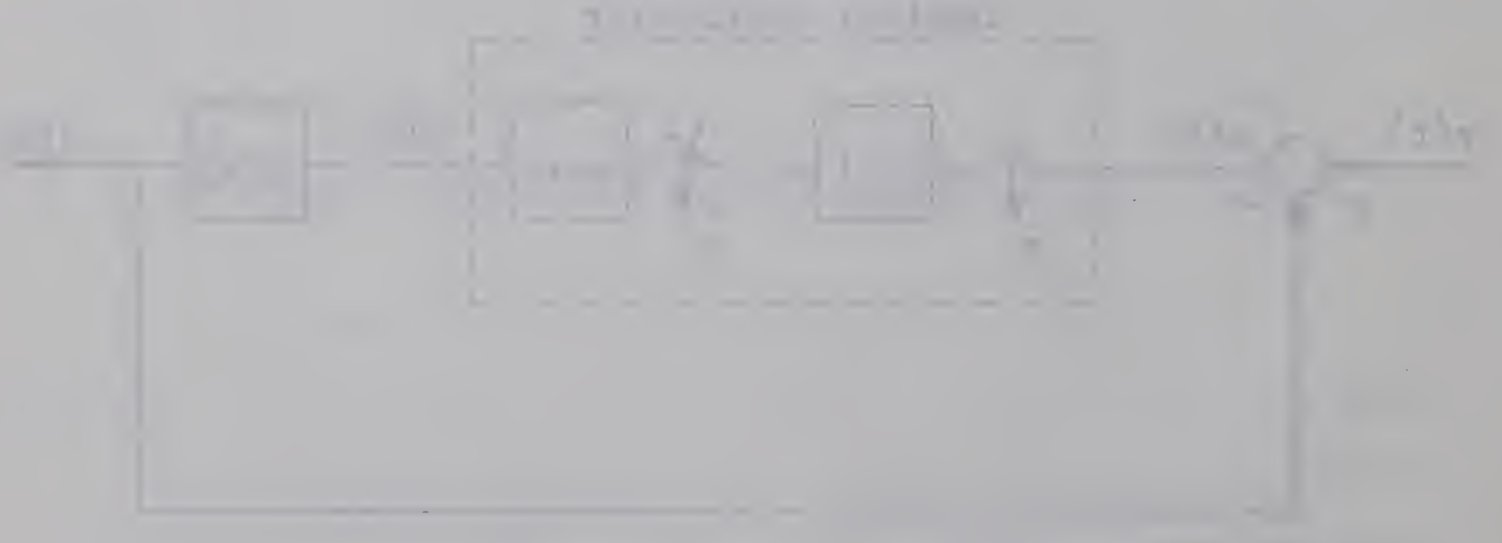


FIGURE 2-2. CONTINUOUS-DATA SYSTEM





consists of the Laplace transform of the controller preceded and followed by synchronous switches and followed by a zero-order hold as shown in Figure 2-2.

2.2 STATE VARIABLES AND STATE TRANSITION EQUATIONS OF CONTINUOUS ELEMENTS

An n-th order, linear, continuous system can be represented by a set of first-order differential equations of the form: $\frac{dx_i}{dt} = f_i(x_1, x_2 \ldots x_n, r_1, r_2 \ldots r_m)$ where the x's are the set of n system variables (state variables); the r's are the m system inputs; the f's are the functional relations.

In vector-matrix notation these equations can be written as

$$\frac{d\bar{x}}{dt} = \bar{A}\bar{x} + \bar{B}\bar{r} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2\text{-}1)$$

where $\bar{x}$ is a column matrix of the state variables and $\bar{r}$ is a column matrix of the inputs. The equation describes the transition of state of the system's state variables from $x_i(t)$ to $x_i(t)+dx_i(t)$ when the time is changed from t to t+dt. The transition is continuous since dt is an infinitesmal time change.

Obtaining the Laplace transform of (2-1) and rearranging the result yields

$$X(s) = (s\bar{I}-\bar{A})^{-1}\bar{x}(t_0^+) + (s\bar{I}-\bar{A})^{-1}\bar{B}\bar{R}(s)$$

where $\bar{x}(t_0^+)$ is a column matrix of the initial conditions of $\bar{x}(t)$ evaluated at $t = t_0^+$. The inverse Laplace transform of X(s) yields

$$\bar{x}(t) = \bar{\Phi}(t-t_0)\bar{x}(t_0^+) + \int_{t_o}^{t} \bar{\Phi}(t-\tau)\bar{B}\bar{r}(\tau)d\tau \quad \ldots\ldots\ldots\ldots\ldots\ldots (2\text{-}2)$$

where $\overline{\Phi}(t-t_0) = e^{(t-t_0)\overline{A}} = \mathcal{L}^{-1}((s\overline{I}-\overline{A})^{-1})$ and the second term on the right-hand side of (2-2) is the convolution integral. This set of equations is defined as the state transition equations and $\overline{\Phi}(t-t_0)$ is referred to as the state transition matrix of the dynamic system.

Using the vector-matrix method provides simplified notations for obtaining a solution, but the signal flow graph approach provides a more straightforward method of writing the state transition equations. The state transition flow graph of a system is the analog computer simulation flow graph with the initial conditions applied as input signals at the corresponding output nodes of the integrators. Using the flow graph and Mason's gain formula[3] the state transition equations can readily be written.

The analog simulation flow graph of the continuous elements may be programmed by three different methods. The two most common methods are direct programming and iterative programming, with the third method being parallel programming. The method one should use is decided by the form of the given transfer function. If the transfer function is not in factored form, direct programming is used, but if it is in factored form or can be factored, then iterative programming is the easiest method to use. Parallel programming is used if the transfer function can be expressed in partial fraction form. Using iterative programming reduces the number of terms in the state transition equations and thus the number of calculations required is reduced. If the calculations

are to be carried out manually, this method not only saves time, but reduces the chances of incurring errors.

To illustrate direct and iterative programming, an example will be developed using both methods. The chosen plant has the transfer function $G(s) = 1/(s^3+3s^2+2s)$ or in factored form $G(s) = 1/s(s+1)(s+2)$.

Using the direct method, the expression for G(s) can be written as

$$(s^3+3s^2+2s)C(s) = R(s) \quad \text{.....(2-3)}$$

where C(s) and R(s) represent the output and input of the plant respectively. Choosing the state variables $X_1(s)$ as

$$X_1(s) = C(s)$$

$$sX_1(s) = X_2(s)$$

$$s^2X_1(s) = X_3(s)$$

$$s^3X_1(s) = X_4(s)$$

and substituting into expression (2-3) yields

$$X_4(s) = R(s) - 3X_3(s) - 2X_2(s).$$

To draw the analog simulation flow graph, the integrators are represented by s^{-1} and X_1, X_2, X_3 are chosen as the nodes. The flow graph is shown in Figure (2-3).

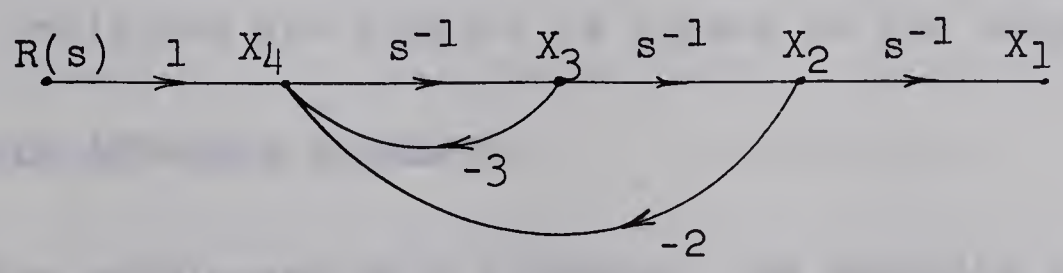
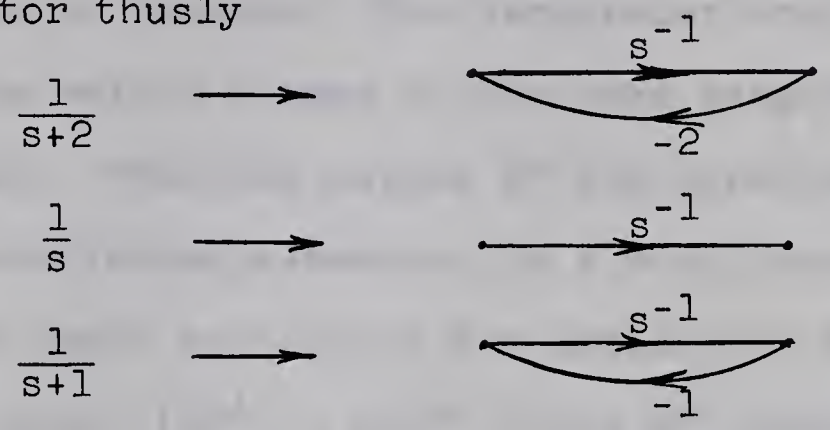


FIGURE 2-3. FLOW GRAPH USING DIRECT METHOD

Using the iterative method, the transfer function is separated into its factors and a separate flow graph is drawn for each factor thusly



To draw the complete flow graph, the separate flow graphs are joined together using unity gain branches as shown in Figure (2-4) with the output nodes of each graph labelled as X_1, X_2, and X_3.

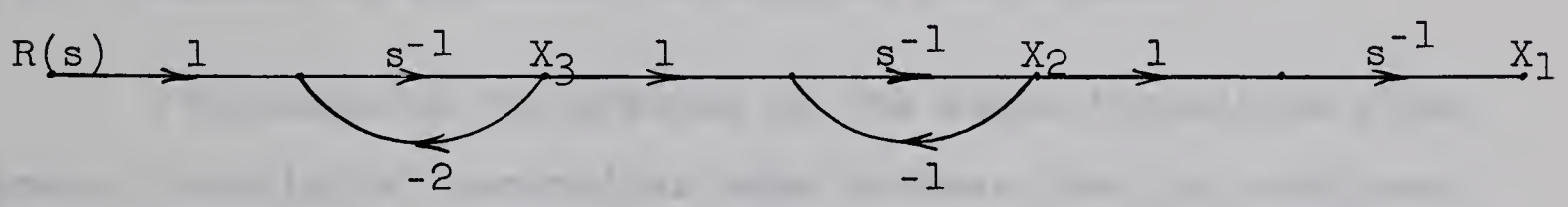
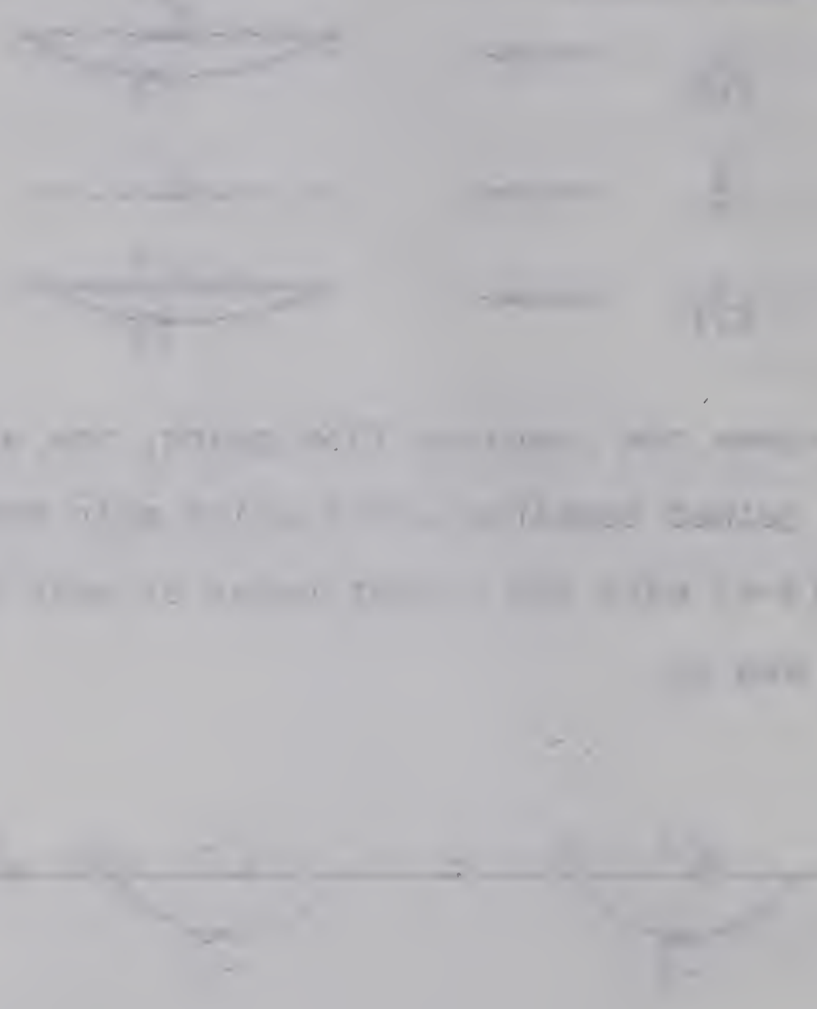


FIGURE 2-4. FLOW GRAPH USING ITERATIVE METHOD

The state transition flow graph can be drawn directly from the analog simulation flow graph. The output nodes of the integrators are chosen as the state variables and the initial conditions are applied as inputs to the output nodes.

2.3 SAMPLE-AND-HOLD ELEMENTS

The sample-and-hold elements, the sampling switch and the zero-order hold, are normally positioned between the digital controller and the continuous elements. The output



of the digital controller is sampled by a switch closed for a short duration, short enough that the input to the zero-order hold appears as an impulse. The zero-order hold retains the value until the switch closes at the next sampling instant T seconds later. Thus the output of the zero-order hold, the input to the continuous elements, is a step function.

If the input and output are denoted by m(t) and h(t) respectively, then $h(kT^+) = m(kT)$ where kT^+ denotes the k-th sampling period from time kT to $kT+\tau$ for $0<\tau\leq T$. The Laplace transform of a step function is 1/s, thus on the state transition flow graph the sample-and-hold elements are represented by s^{-1}.

2.4 VARIABLE GAIN CONCEPT OF DIGITAL CONTROLLERS

To complete the drawing of the state transition flow graph, the digital controller must be described by a mathematical expression. The variable gain concept will be employed for this purpose.

Referring to Figure 2-1, the output of the zero-order hold is h(t) and the controller actuating signal is e(t). The digital controller output is a train of impulses with values at the sampling instants which are equal to the zero-order hold output at the sampling instants. Under this condition the z-transfer function can be written as

$$D(z) = H(z)/E(z)$$

where $H(z) = h(0^+) + h(T^+)z^{-1} + h(2T^+)z^{-2} + \ldots + h(nT^+)z^{-n}$ and

$E(z) = e(0^+) + e(T^+)z^{-1} + e(2T^+)z^{-2} + \ldots + e(nT^+)z^{-n}$.

Using the state variable approach, the digital controller may be replaced by a variable gain amplifier with gain K(kT). The amplifier will have different values during different sampling periods since during any one sampling period only one term in each of the equations for H(z) and E(z) is of interest. The gain during the first sampling period is given by $K(0) = h(0^+)/e(0^+)$, the second sampling period by $K(T) = h(1T^+)/e(1T^+)$ and so on. Therefore the gain during the (k+1)-th sampling period is given by $K(kT) = h(kT^+)/e(kT^+)$. Thus, when drawing the state transition flow graph for the time interval $kT < t \leq (k+1)T$, the digital controller will be represented by the variable gain amplifier with gain K(kT).

2.5 STATE TRANSITION FLOW GRAPH

With the preceding element descriptions, the complete state transition flow graph may be drawn. As previously mentioned, to represent the continuous elements, the analog simulation flow graph is modified by adding initial conditions to the output nodes of the integrators. The initial conditions are represented by a unity gain branch from $x(t_0)$ to $x(t_0^+)$ and from $x(t_0^+)$ to the state variable X(s) by a branch with gain 1/s. The reason for this is that the initial conditions appear initially as step functions at the outputs of the integrators.

To illustrate the final flow graph an example will be described using the closed-loop system of Figure 2-5. The analog simulation flow graph of G(s) is shown in Figure 2-4.

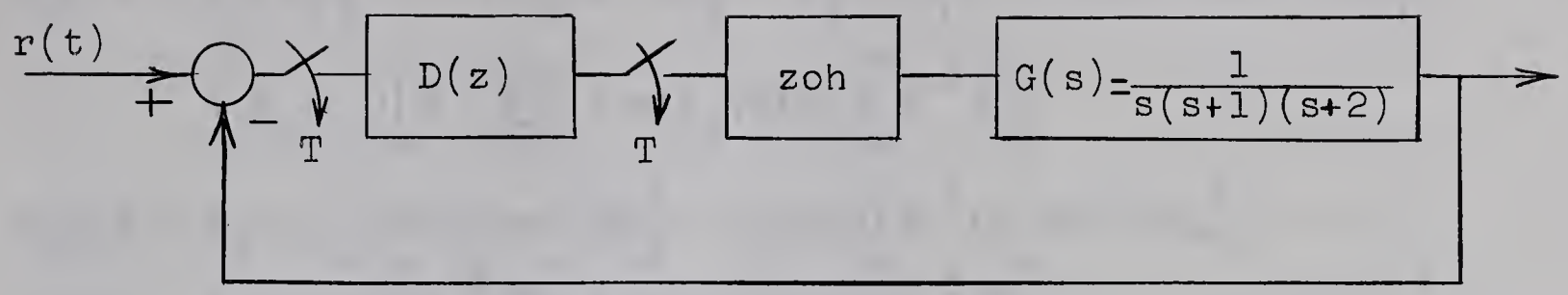


FIGURE 2-5. CLOSED-LOOP SYSTEM

Combining this with the variable gain concept of the digital controller and the sample-and-hold element representation, the complete state transition flow graph may be drawn as shown in Figure 2-6. The digital controller is described by $K(t_0)$.

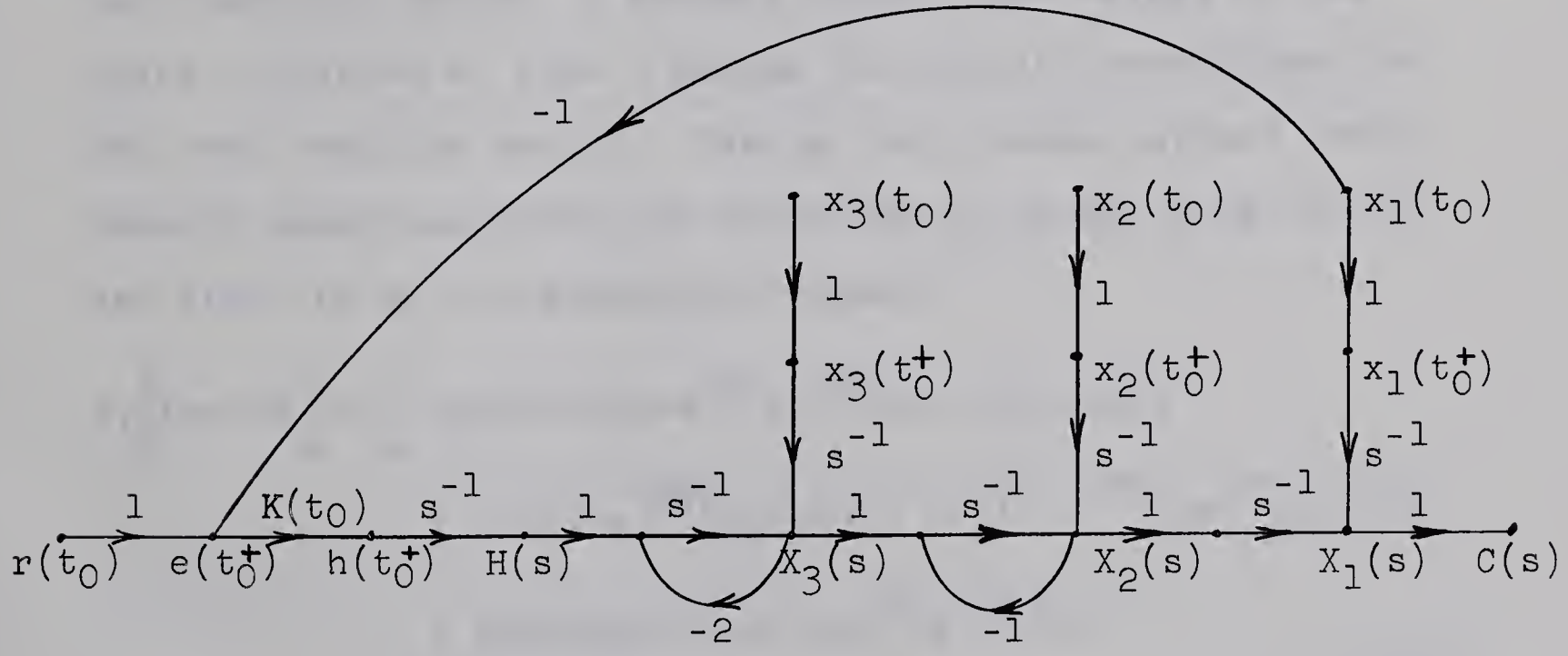
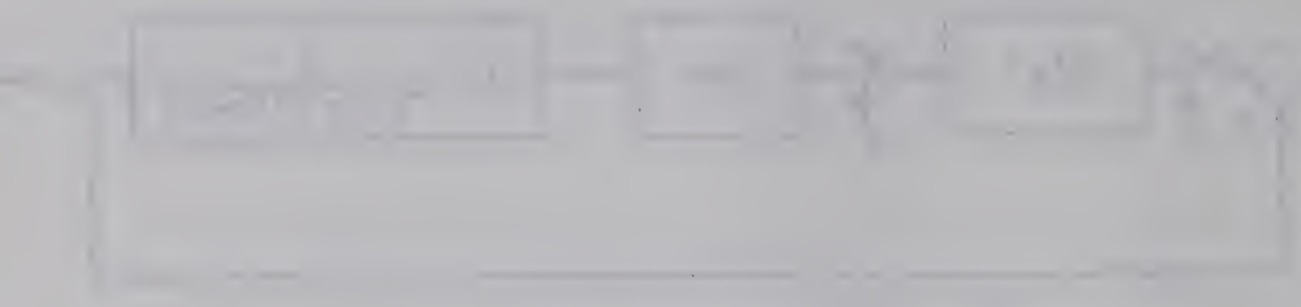


FIGURE 2-6. STATE TRANSITION FLOW GRAPH

2.6 STATE TRANSITION EQUATIONS

The state transition equations can be written directly from the flow graph using Mason's gain formula[3]. The equations are:



$$X_1(s) = x_1(t_0)\left[1/s-K(t_0)s^{-4}/\Delta\right] + x_2(t_0)\left[s^{-2}(1+3s^{-1}+2s^{-2})/\Delta\right]$$
$$+ x_3(t_0)\left[s^{-3}/\Delta\right] + r(t_0)K(t_0)\left[s^{-4}/\Delta\right]$$
$$X_2(s) = x_1(t_0)\left[-K(t_0)s^{-3}/\Delta\right] + x_2(t_0)\left[s^{-1}(1-2s^{-1})/\Delta\right]$$
$$+ x_3(t_0)\left[s^{-2}/\Delta\right] + r(t_0)K(t_0)\left[s^{-3}/\Delta\right] \qquad .....(2\text{-}4)$$
$$X_3(s) = x_1(t_0)\left[-K(t_0)s^{-2}/\Delta\right] + x_3(t_0)\left[s^{-1}/\Delta\right]$$
$$+ r(t_0)K(t_0)\left[s^{-2}/\Delta\right]$$

where $\Delta = (1+3s^{-1}+2s^{-2})$.

The digital controller starts operating at time t_0=0 seconds and the output of the plant is continuous until the next sampling period, T seconds later. The values of the state variables at time T become the initial conditions for the next sampling period. Taking the inverse Laplace transform of equations (2-4) and replacing t_0 by kT, t by (k+1)T and K(kT) by K_k the equations become

$$x_1\left[(k+1)T\right] = \left[-K_k(T/2-3/4+e^{-T}-e^{-2T}/4) + 1\right]x_1(kT)$$
$$+ 0.5(1-e^{-2T})x_2(kT) + 0.5(1+e^{-2T}-2e^{-T})x_3(kT)$$
$$+ r(kT)K_k(T/2-3/4+e^{-T}-e^{-2T}/4) \qquad .....(2\text{-}5)$$
$$x_2\left[(k+1)T\right] = -0.5K_k(1+e^{-2T}-2e^{-T})x_1(kT) + (e^{-2T})x_2(kT)$$
$$+ (e^{-T}-e^{-2T})x_3(kT) + 0.5K_k(1+e^{-2T}-2e^{-T})r(kT)$$
$$x_3\left[(k+1)T\right] = -K_k(1-e^{-T})x_1(kT) + (e^{-T})x_3(kT) + (1-e^{-T})K_k r(kT)$$

For deadbeat response, the system error must be zero

for $t \geq nT$ where n is the smallest possible positive integer. For a third-order system, three is the smallest value n can have. The reason for this is that there are three conditions which must be satisfied, the values of the three state variables. The state variables are functions of the variable gain K_k where k takes on the values 0, 1 and 2. Letting k take on these values produces three equations in three unknowns. Applying a unit step input to the above system, the output of the last integrator (x_1) would be unity for the system to have zero steady state error. In order that this output does not change, the remaining state variables (x_2 and x_3) must be zero at the end of the third sampling period.

Proceeding with the solution for a unit step input and a sampling period of T= 1 second, the equations become

$$\begin{aligned} x_1\left[(k+1)T\right] &= (-.084046K_k + 1)x_1(kT) + (.432332)x_2(kT) \\ &\quad + (.199786)x_3(kT) + (.084046)K_k \\ x_2\left[(k+1)T\right] &= (-.199786K_k)x_1(kT) + (.135335)x_2(kT) \\ &\quad + (.23254)x_3(kT) + (.199786)K_k \qquad .....(2\text{-}6) \\ x_3\left[(k+1)T\right] &= (-.63212K_k)x_1(kT) + (.36788)x_3(kT) \\ &\quad + (.63212)K_k \end{aligned}$$

Letting k=0 in (2-6) yields the values of the state variables at the end of the first sampling period which are

$$\begin{aligned} x_1(T) &= .084046K_0 \\ x_2(T) &= .199786K_0 \qquad .....(2\text{-}7) \\ x_3(T) &= .63212K_0 \end{aligned}$$

Setting k=1 in (2-6) and substituting (2-7) into (2-6) results in the equations

$$x_1(2T) = -.00706373K_0K_1 + .084046K_1 + .296709K_0$$
$$x_2(2T) = -.0167912K_0K_1 + .199786K_1 + .174031K_0 \quad .....(2\text{-}8)$$
$$x_3(2T) = -.0531272K_0K_1 + .63212K_1 + .232544K_0$$

Making a final substitution of k=2 in the equations (2-6) and substituting the expressions (2-8) into (2-6) produces the required equations which are

$$x_1(3T) = .000593678K_0K_1K_2 - .00706373K_2K_1 - .0249372K_0K_2$$
$$-.0249372K_0K_1 + .084046K_2 + .296709K_1 + .418407K_0 \quad ....(2\text{-}9)$$

$$x_2(3T) = .00141123K_0K_1K_2 - .0167912K_2K_1 - .0592783K_0K_2$$
$$- .0146266K_0K_1 + .199786K_2 + .174031K_1 + .0776283K_0 \quad ....(2\text{-}10)$$

$$x_3(3T) = .00446513K_0K_1K_2 - .0531272K_1K_2 - .187556K_0K_2$$
$$-.0195444K_0K_1 + .232544K_1 + .63212K_2 + .0855483K_0 \quad ....(2\text{-}11)$$

For deadbeat response $x_1(3T) = 1$, $x_2(3T) = 0$, and $x_3(3T) = 0$. There are now three simultaneous equations in the three unknowns K_0, K_1 and K_2. These equations are solved by eliminating the unknowns in a particular sequence, starting with the gain having the largest subscript. That is, multiplying by the coefficients of the K_2 terms in a third-order system and subtracting the result, eliminates all terms containing K_2 from the equations. Continuing, choosing the coefficients of the K_1 terms, multiplying and subtracting, eliminates all terms with the factor K_1, leaving one equation

in the remaining unknown K_0. Beginning the elimination of unknowns with the coefficients of a term other than K_2 will not eliminate all the product terms containing that factor.

For example, multiplying (2-10) by .63212 and (2-11) by 0.199786 and subtracting the result yields

$$-.00534107K_0K_1 + .063549K_1 + .031979K_0 = 0 \quad .....(2\text{-}12)$$

in which all terms containing the factor K_2 have been eliminated. Continuing the solution in this manner results in the required variable gains

$$\begin{aligned} K_0 &= 3.659199 \\ K_1 &= -2.659186 \\ K_2 &= 2.638810 \end{aligned} \qquad .....(2\text{-}13)$$

Having obtained the variable gains the next step is to determine the transfer function of the digital controller.

2.7 TRANSFER FUNCTION OF DIGITAL CONTROLLER

To determine the transfer function, the error functions, that is, inputs to the digital controller, must be calculated at the sampling instants. The input $r(kT)$ is a unit step and the output can be determined from the state variable equations (2-7) and (2-8) for $x_1(T)$ and $x_1(2T)$ respectively. The output is considered to be initially zero, therefore the error functions are

$$\begin{aligned} e(0^+) &= r(0) - x_1(0) = 1 \\ e(T^+) &= r(T) - x_1(T) = .692459 \\ e(2T^+) &= r(2T) - x_1(2T) = .069043 \end{aligned}$$

The outputs of the digital controller are found from the variable gain equation $K_k = h(kT^+)/e(kT^+)$ and are calculated to be

$$h(0^+) = K_0 e(0^+) = 3.659199$$
$$h(T^+) = K_1 e(T^+) = -1.841377$$
$$h(2T^+) = K_2 e(2T^+) = 0.182191$$

The complete transfer function can now be written

$$D(z) = \frac{3.659199 - 1.841377z^{-1} + 0.182191z^{-2}}{1 + 0.692459z^{-1} + 0.069043z^{-2}}$$

The z-transform is used since the complete system is a sampled-data system and only one term is valid during each sampling period.

The digital controller can be realized on the analog computer with the aid of time delays or on a digital computer using data storage[4].

As can be seen from the above example, the numbers involved in the solution are cumbersome when solving the equations on a calculator and the possibilities of incurring errors are numerous. To obtain an accurate answer quickly a digital computer is required. The next chapter is devoted to a FORTRAN program for use on the IBM 7040 computer.

CHAPTER III

PROGRAM FOR SOLVING STATE EQUATIONS

The state transition approach to calculating the transfer function of a digital controller, which will produce deadbeat response when used as a compensator, lends itself to solution on a digital computer. All calculations are step-by-step multiplications, subtractions and divisions as can be seen from the example in Chapter II. The program to be described in this chapter is not a general program, but one written to solve specific types of systems.

3.1 SYSTEMS SOLVABLE USING DIGITAL PROGRAM

The program was designed for use with linear third- and fourth-order systems. The transfer functions of these systems must not contain any zeros and must have at least one pole at the origin, that is, a free integrator. A further restriction is that the excitation function be a step function. For calculation purposes, the value of the step input can be assumed to be unity since the system is linear and continuous between sampling instants.

The system flow graph must be drawn using the iterative or direct methods. If the partial-fraction method of drawing the flow graph were used then the program would not provide a solution. The reason will be explained in a later section.

3.2 DIGITAL PROGRAM FOR USE ON THE IBM 7040 COMPUTER

The program, written in FORTRAN IV computing language,

performs each calculation carried out in the solution of the example described in Chapter II.

The coefficients of the state variables are placed in an E or F matrix depending on whether the system is of third- or fourth-order respectively. Using the example of Chapter II, the coefficients of equations (2-5) would be placed in the first three rows of the E matrix. Making the appropriate substitutions and calculations, the values of the state variables in terms of the variable gains are obtained at the sampling instants. The coefficients of the variable gains and products of variable gains, as shown in equations (2-7) to (2-11), are stored in succeeding rows of the matrix. The final three rows of the matrix contain the coefficients of equations (2-9) to (2-11) which are the three equations in the three unknowns (the variable gains) that have to be solved. To solve these three equations, their coefficients are first transferred to an A matrix, to a B matrix if the system is of fourth-order, where the unknowns are eliminated and the coefficients of the reduced equations are stored.

The final row of the matrix A contains the coefficients of the equation in the unknown K_0. The variable gains are then solved for along with the values of the errors at the sampling instants and the values of the inputs to the plant.

The program shown below contains a number of statements precededed by a "C", which indicates the statement is a comment. Comment statements do not affect the execution of the program, and are used to indicate what calculations are to follow.

Preceding the program is a list of the terms used in the program.

CKO, CK1, CK2, CK3 ----- Variable gains K_0, K_1, K_2, K_3

CKN2 -----Numerator of expression for CK2

CKD2 -----Denominator of expression for CK2

ER0 ------error at time $t=0^+$

ER1 ------error at time $t=T^+$

ER2 ------error at time $t=2T^+$

H0 -------input to plant at time $t=0^+$

H1 -------input to plant at time $t=T^+$

H2 -------input to plant at time $t=2T^+$

T --------sampling period in seconds

N --------order of plant

The flow chart of the digital program shown below is shown in Appendix A.

```
$IBFTC EQNS
C  SOLUTION OF STATE EQUATIONS TO OBTAIN THE EXPRESSION
C  FOR A DIGITAL CONTROLLER
      COMMON A(12,8), B(13,16), E(12,7), F(17,15)
      READ (5,1) N,T
    1 FORMAT (I5,E12.3)
      WRITE (6,2) N,T
    2 FORMAT(1X,I5,E10.3)
      CALL ARRAYS (N,T)
      IF (N.EQ.3) GO TO 49
C  OBTAIN COEFFICIENTS OF FOURTH-ORDER STATE VARIABLE EQUATIONS
```

```
      F(5,1) = F(1,6)
      DO 3 J = 2,N
    3 F(5,J) = F(J,5)
      DO 4 I = 1,N
    4 F(I+5,1) = F(I,1)*F(5,1)
      F(6,2) = F(1,6)
      DO 5 I = 2,N
    5 F(I+5,2) = F(I,5)
      F(6,3)=F(1,2)*F(5,1)+F(1,3)*F(5,2)+F(1,4)*F(5,3)+F(1,5)*
     1F(5,4)
      DO 7 I=2,N
    7 F(I+5,3)=F(I,2)*F(5,2)+F(I,3)*F(5,3)+F(I,4)*F(5,4)
      DO 9 I=1,N
      DO 9 J=1,3
    9 F(I+9,J)=F(I,1)*F(6,J)
      DO 11 I=1,3
   11 F(10,I+3)=F(1,2)*F(6,I)+F(1,3)*F(7,I)+F(1,4)*F(8,I)+
     1F(1,5)*F(9,I)
      F(10,7)=F(1,6)
      DO 13 I=1,3
      DO 13 J=1,3
   13 F(I+10,J+3)=F(I+1,2)*F(7,J)+F(I+1,3)*F(8,J)+F(I+1,4)*
     1F(9,J)
      DO 15 I=1,3
   15 F(I+10,7)=F(I+1,5)
      DO 17 I=1,4
      DO 17 J=1,7
   17 F(I+13,J)=F(I,1)*F(10,J)
```

```
      DO 19 I=1,4
   19 F(I+13,8)=F(I+9,7)
      DO 21 I=1,7
   21 F(14,I+8)=F(1,2)*F(10,I)+F(1,3)*F(11,I)+F(1,4)*F(12,I)+
     1F(1,5)*F(13,I)
      DO 23 I=1,3
      DO 23 J=1,7
   23 F(I+14,J+8)=F(I+1,2)*F(11,J)+F(I+1,3)*F(12,J)+F(I+1,4)*
     1F(13,J)
C  TRANSFER THE COEFFICIENTS TO B MATRIX
      DO 40 I=1,N
      DO 40 J=1,15
   40 B(I,J)=F(I+13,J)
C  ELIMINATE VARIABLE GAIN CK3
      DO 42 I=1,3
      DO 42 J=1,16
   42 B(I+4,J)=B(I+1,J)*B(1,8)
      DO 44 I=2,4
      DO 44 J=1,16
   44 B(I+6,J)=B(1,J)*B(I,8)
      DO 46 I=1,3
      DO 46 J=1,16
   46 B(I+10,J)=B(I+4,J)-B(I+7,J)
C  TRANSFER COEFFICIENTS TO MATRIX A TO SOLVE EQUATIONS
      DO 48 I=1,3
      DO 48 J=1,8
   48 A(I,J)=B(I+10,J+8)
      GO TO 80
```

```
C  MANIPULATION OF E MATRIX TO OBTAIN COEFFICIENTS OF THIRD-
C  ORDER STATE VARIABLE EQUATIONS
   49 E(4,1)=E(1,5)
      E(5,1)=E(2,4)
      E(6,1)=E(3,4)
      DO 50 I=1,3
   50 E(I+6,1)=E(I,1)*E(4,1)
      DO 52 I=1,3
   52 E(I+6,2)=E(I+3,1)
      E(7,3)=E(1,2)*E(4,1)+E(1,3)*E(5,1)+E(1,4)*E(6,1)
      DO 54 I=1,2
   54 E(I+7,3)=E(I+1,2)*E(5,1)+E(I+1,3)*E(6,1)
      DO 56 I=1,3
      DO 56 J=1,3
   56 E(I+9,J)=E(I,1)*E(7,J)
      DO 58 I=1,3
   58 E(10,I+3)=E(1,2)*E(7,I)+E(1,3)*E(8,I)+E(1,4)*E(9,I)
      DO 60 I=1,3
   60 E(I+9,7)=E(I+3,1)
      DO 62 I=1,2
      DO 62 J=1,3
   62 E(I+10,J+3)=E(I+1,2)*E(8,J)+E(I+1,3)*E(9,J)
      DO 66 I=1,N
      DO 66 J=1,7
   66 A(I,J)=E(I+9,J)
C  SOLUTION OF THIRD-ORDER SIMULTANEOUS EQUATIONS
   80 DO 82 I=1,2
      DO 82 J=1,8
```

```
82 A(I+3,J)=A(I+1,J)*A(1,7)
   DO 84 I=2,3
   DO 84 J=1,8
84 A(I+4,J)=A(1,J)*A(I,7)
   DO 86 I=1,2
   DO 86 J=1,8
86 A(I+7,J)=A(I+3,J)-A(I+5,J)
   DO 87 J=1,8
87 A(10,J)=A(8,J)*A(9,5)
   DO 88 J=1,8
88 A(11,J)=A(9,J)*A(8,5)
   DO 89 J=1,8
89 A(12,J)=A(10,J)-A(11,J)
C  SOLVE FOR VARIABLE GAINS
     CKO=A(12,8)/A(12,6)
     CK1=(A(8,8)-A(8,6)*CKO)/(A(8,4)*CKO-A(8,5))
     CNK2=A(1,8)-A(1,4)*CKO*CK1-A(1,5)*CK1-A(1,6)*CKO
     CDK2=A(1,1)*CKO*CK1+A(1,2)*CK1+A(1,3)*CKO+A(1,7)
     CK2=CNK2/CDK2
     IF (N.EQ.3) GO TO 91
     CKO1=CKO*CK1
     CKO2=CKO*CK2
     CK12=CK1*CK2
     CKO12=CKO*CK1*CK2
     CKN3=B(1,16)-B(1,9)*CKO12-B(1,10)*CK12-B(1,11)*CKO2-
    1B(1,12)*CKO1-B(1,13)*CK1-B(1,14)*CKO-B(1,15)*CK2
     CKD3=B(1,1)*CKO12+B(1,2)*CK12+B(1,3)*CKO2+B(1,4)*CKO1+
    1B(1,5)*CK1+B(1,6)*CKO+B(1,7)*CK2+B(1,8)
```

```
      CK3=CKN3/CKD3
      WRITE (6,90) CKO,CK1,CK2,CK3
   90 FORMAT (26H CONTROLLER GAINS CKO--CK3/4E16.8)
      GO TO 97
   91 WRITE (6,92) CKO,CK1,CK2
   92 FORMAT (26H CONTROLLER GAINS CKO--CK2/3E16.8)
C  D(Z) FOR STEP INPUTS
C  ERRORS FOR THIRD-ORDER SYSTEM
      ERO=A(1,8)
      ER1=A(1,8)-E(4,1)*CKO
      ER2=A(1,8)-E(7,1)*CKO*CK1-E(7,2)*CK1-E(7,3)*CKO
C  PLANT INPUTS FOR THIRD-ORDER SYSTEM
      HO=CKO*ERO
      H1=CK1*ER1
      H2=CK2*ER2
      WRITE (6,94) ERO,ER1,ER2
   94 FORMAT (23H ERROR SIGNALS ERO--ER2/3E16.8)
      WRITE (6,95) HO,H1,H2
   95 FORMAT (26H CONTROLLER OUTPUTS HO--H2/3E16.8)
      CALL EXIT
C  ERRORS FOR FOURTH-ORDER SYSTEM
   97 ERO=B(1,16)
      ER1=B(1,16)-F(5,1)*CKO
      ER2=B(1,16)-F(6,1)*CKO*CK1-F(6,2)*CK1-F(6,3)*CKO
      ER3=B(1,16)-F(10,1)*CKO12-F(10,2)*CK12-F(10,3)*CKO2-
     1F(10,4)*CKO1-F(10,5)*CK1-F(10,6)*CKO-F(10,7)*CK2
C  PLANT INPUTS FOR FOURTH-ORDER SYSTEM
      HO=CKO*ERO
```

```
      H1=CK1*ER1
      H2=CK2*ER2
      H3=CK3*ER3
      WRITE (6,98) ER0,ER1,ER2,ER3
   98 FORMAT (23H ERROR SIGNALS ER0--ER3/4E16.8)
      WRITE (6,99) H0,H1,H2,H3
   99 FORMAT (26H CONTROLLER OUTPUTS H0--H3/4E16.8)
      END

$IBFTC COEF
      SUBROUTINE ARRAYS (N,T)
      COMMON A(12,8),B(13,16),E(12,7),F(17,15)
C  THIS SUBROUTINE IS USED TO PUNCH THE COEFFICIENTS OF THE
C  STATE VARIABLES AND THE FINAL VALUES OF THE STATE EQUATIONS
C  INTO THEIR APPROPRIATE MATRICES.  FOR EXAMPLE, THE VALUES
C  IMMEDIATELY FOLLOWING ARE THE REQUIRED DATA FOR THE PLANT
C  G(S)=1/S(S+1)(S+2).
      A(1,8)=1.0
      A(2,8)=0.0
      A(3,8)=0.0
      E(1,1)=-(T/2.-3./4.+EXP(-T)-0.25*EXP(-2.*T))
      E(1,2)=1.
      E(1,3)=0.5-0.5*EXP(-2.*)
      E(1,4)=0.5+0.5*EXP(-2.*T)-EXP(-T)
      E(1,5)=T/2.-0.75+EXP(-T)-0.25*EXP(-2.*T)
      E(2,1)=-0.5-0.5*EXP(-2.*T)+EXP(-T)
      E(2,2)=EXP(-2.*T)
      E(2,3)=EXP(-T)-EXP(-2.*T)
```

```
      E(2,4)=0.5+0.5*EXP(-2.*T)-EXP(-T)
      E(3,1)=-1.+EXP(-T)
      E(3,2)=0.0
      E(3,3)=EXP(-T)
      E(3,4)=1.-EXP(-T)
      RETURN
      END
```

3.3 INPUT

The coefficients of the state variables, in terms of the sampling period T, are introduced into the program using the subroutine "ARRAYS". This allows T to be varied and the magnitudes of the resulting gains to be examined. Each coefficient is placed in a specific location in the E or F matrix. The state variable equations for a third-order system with the coefficients replaced by the appropriate matrix designations are shown below.

$$x_1\left[(k+1)T\right] = \left[E(1,1)K_k + E(1,2)\right]x_1(kT) + E(1,3)x_2(kT) + E(1,4)x_3(kT) + E(1,5)K_k \quad .....(3\text{-}1)$$

$$x_2\left[(k+1)T\right] = E(2,1)K_k x_1(kT) + E(2,2)x_2(kT) + E(2,3)x_3(kT) + E(2,4)K_k \quad .....(3\text{-}2)$$

$$x_3\left[(k+1)T\right] = E(3,1)K_k x_1(kT) + E(3,2)x_2(kT) + E(3,3)x_3(kT) + E(3,4)K_k \quad .....(3\text{-}3)$$

The coefficients of a fourth-order system are similarly read into the F matrix locations.

The predetermined final values, to which the state equations are equated, are placed in the A or B matrix as follows:

for a third-order system

A(1,8)= value of x_1 at time 3T

A(2,8)= value of x_2 at time 3T

A(3,8)= value of x_3 at time 3T

for a fourth-order system

B(1,16)= value of x_1 at time 4T

B(2,16)= value of x_2 at time 4T

B(3,16)= value of x_3 at time 4T

B(4,16)= value of x_4 at time 4T

To complete the input information, the numerical values of the order (N) of the system and the sampling period (T seconds) are punched on data cards. The last data card has a 9 punched into the location for system order. When the computer reads this card, the execution of the program is stopped.

3.4 OUTPUT

Shown below is a sample print-out of the program for the plant with transfer function G(s)=1/s(s+1)(s+2).

```
     3 0.100E 01
CONTROLLER GAINS CKO--CK2
 0.36591675E 01 -0.26591258E 01  0.26386268E 01
ERROR SIGNALS ERO--ER2
 0.10000000E 01  0.69246299E 00  0.69042742E-01
```

CONTROLLER OUTPUTS HO--H2

0.36591675E 01 -0.18413462E 01 0.18217802E 00

The first line of the output is a print-out of the input data card. The integer 3 is the order of the system and the exponential number 0.100E 01 is the sampling period in seconds, one second in this case.

The CONTROLLER GAINS CKO--CK2 are the variable gains

$$K_0 = 3.6591675$$

$$K_1 = -2.6591257$$

$$K_2 = 2.6386267$$

The ERROR SIGNALS ERO--ER2 are the errors at the sampling instants

$$e(0^+) = 1.00000000$$

$$e(T^+) = 0.69246299$$

$$e(2T^+) = 0.069042742$$

The CONTROLLER OUTPUTS HO--H2 are the inputs to the plant

$$h(0^+) = 3.6591675$$

$$h(T^+) = -1.8413462$$

$$h(2T^+) = 0.1817802$$

CHAPTER IV

TRANSFER FUNCTIONS OF DEADBEAT CONTROLLERS

Using the program described in the previous chapter, the transfer functions of deadbeat controllers for various systems will be obtained and the results confirmed on the PACE 231-R analog computer. The procedure used to solve a system containing a zero in the transfer function will also be described.

4.1 THIRD-ORDER SYSTEM WITH SIMPLE POLES

For a plant, which has a transfer function consisting of simple poles only, the deadbeat response that is obtained has no overshoot irrespective of the sampling period. But, the sampling period does affect the magnitudes of the variable gains. Considering the plant $G(s)= 1/s(s+1)(s+2)$ and its flow graph shown in Figure 2-6, solving for sampling periods T=1 second and T=0.1 second, yields the transforms

$$\text{T=1 second} \quad D(z)=\frac{3.6591675-1.8413462z^{-1}+0.18217802z^{-2}}{1.0000000+0.69246299z^{-1}+0.06904274z^{-2}}$$

$$\text{T=0.1 second} \quad D(z)=\frac{1159.4171-1998.3344z^{-1}+858.91718z^{-2}}{1.0000000+0.82059483z^{-1}+0.15441565z^{-2}}$$

The validity of the values for T=1 second was verified on the analog computer using the rep-op method*. The analog

---

* A description of this method, developed by Professor Y. J. Kingma, may be found in Appendix A of the master's thesis by Sushil K. Sarna on 'Sampled-Data Conditional Feedback Systems', University of Alberta, September, 1966.

Statutory Provisions on Statutory Construction

Using the chapter heading as the governing subject, the investor identifies at least one relevant statutory provision that may be analyzed and the legal principle contained in the [illegible] analysis [illegible]. The discussion ends by noting [illegible] further investigating a case in the relevant jurisdiction to determine [illegible] be construed.

A. Short-Term Investment Plans

The [illegible] begins by a question [illegible] statutory or common [illegible] the [illegible] [illegible] the [illegible] [illegible] [illegible]

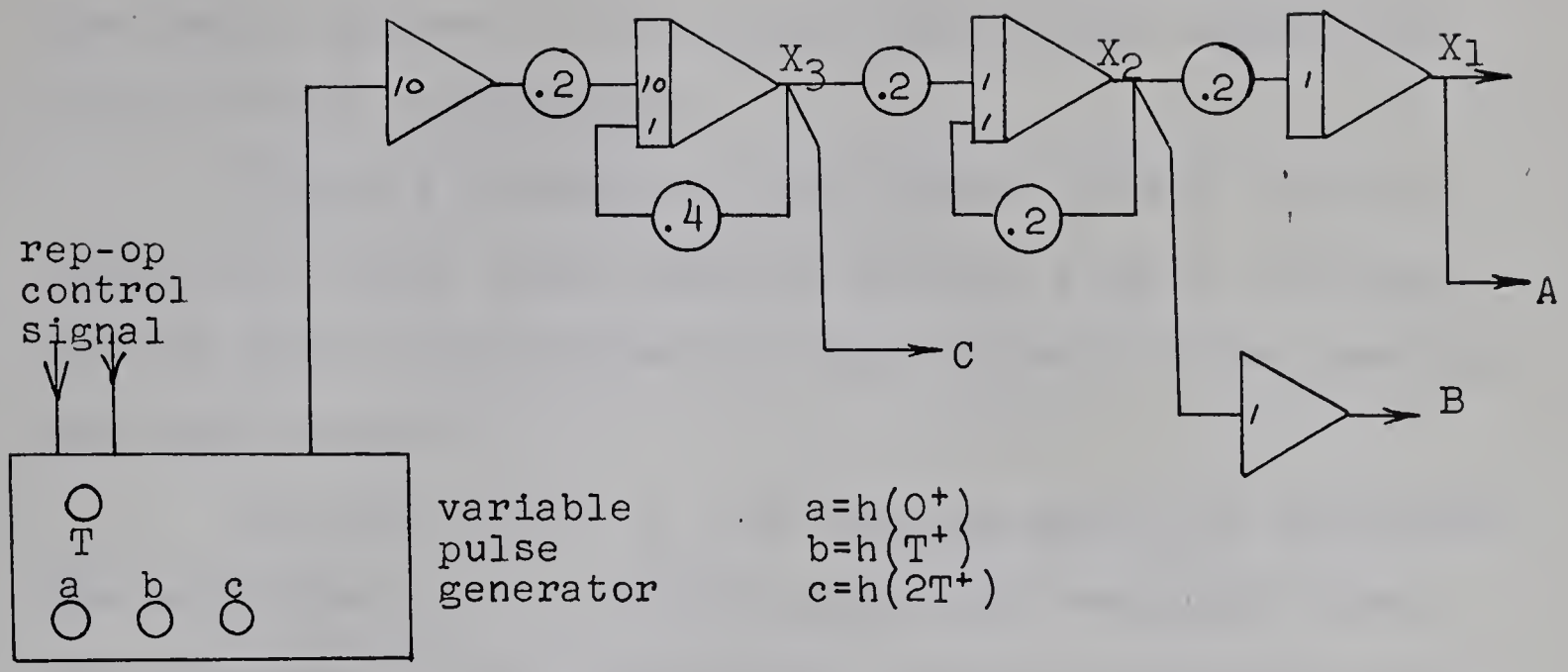


FIGURE 4-1. ANALOG SIMULATION FLOW GRAPH FOR $G(s)=\frac{1}{s(s+1)(s+2)}$

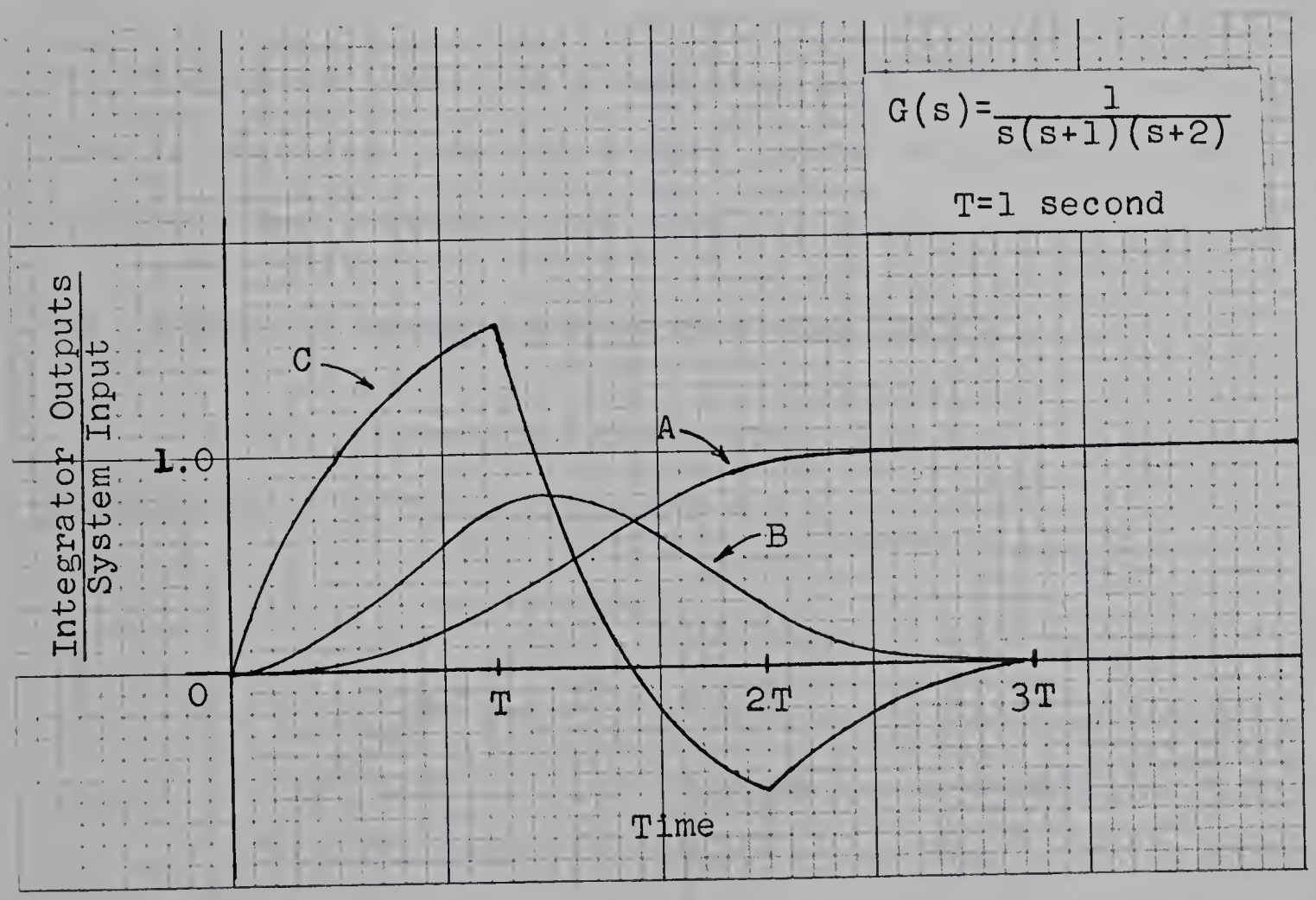


FIGURE 4-2. STATE VARIABLES OF COMPENSATED SYSTEM

diagram is shown in Figure 4-1 and the results in Figure 4-2. The analog diagram is time-scaled for optimum display on a multi-channel oscilloscope.

Figure 4-2 shows that the output (A) has reached a final value after three sampling periods with no overshoot and the other state variables (B and C) have become zero at the same instant.

As the duration of the sampling period is decreased, the magnitudes of the variable gains and the plant inputs are increased. This is expected, as the output must rise to its final value in a shorter time. The z-transform of the digital controller, for a sampling period of T=0.1 seconds, exhibits the large variable gains and plant inputs required as compared to those for a sampling period of T=1.0 seconds. Thus in practice, the gains that can be realized by control equipment may determine the sampling period.

4.2 EFFECT OF REARRANGING SYSTEM FLOW GRAPHS

Using an example, the system flow graph of Figure 2-6 is rearranged to that of Figure 4-3.

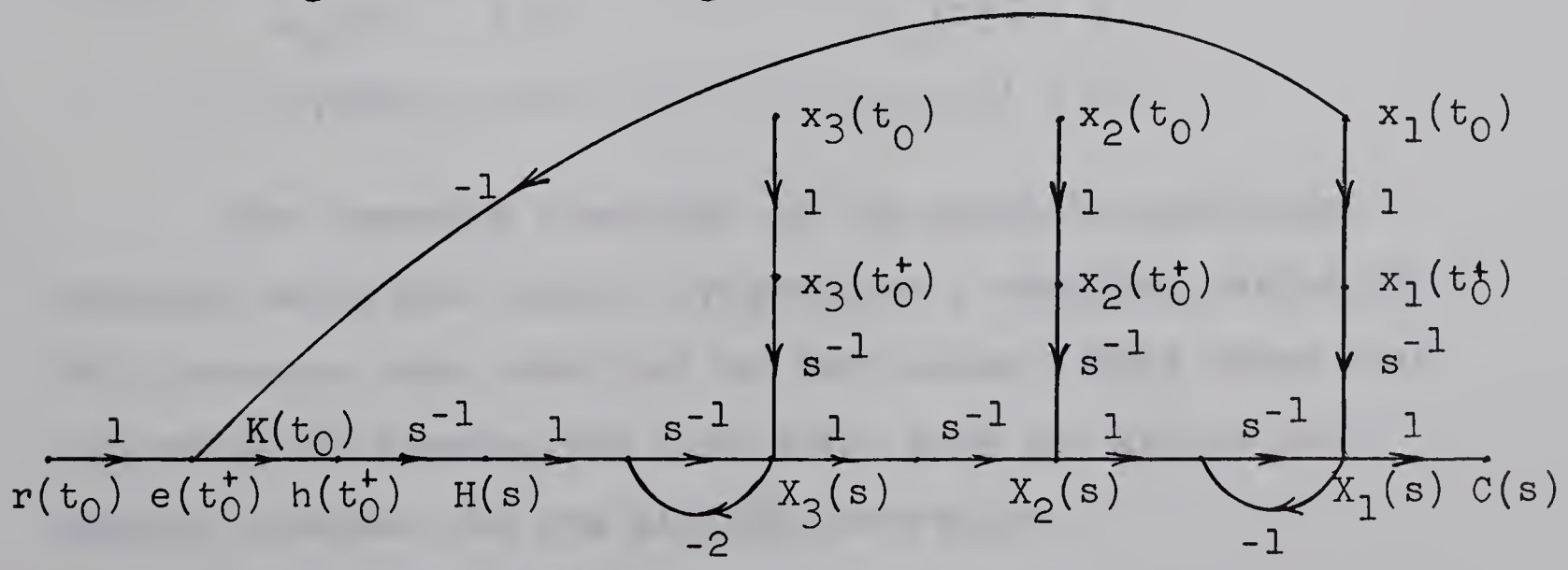


FIGURE 4-3. STATE TRANSITION FLOW GRAPH

The state equations written from the flow graph of Figure 4-3 are

$$X_1(s) = \left[-K_k s^{-4}/\Delta + s^{-1}\right]x_1(t_0) + \left[s^{-2}(1+2s^{-1})/\Delta\right]x_2(t_0) + \left[s^{-3}/\Delta\right]x_3(t_0) + r(t_0)K_k\left[s^{-4}/\Delta\right]$$

$$X_2(s) = \left[-K_k s^{-3}(1+s^{-1})/\Delta\right]x_1(t_0) + \left[s^{-1}\right]x_2(t_0) + \left[s^{-2}(1+s^{-1})/\Delta\right]x_3(t_0) + r(t_0)K_k\left[s^{-3}(1+s^{-1})/\Delta\right] \quad .....(4\text{-}1)$$

$$X_3(s) = \left[-K_k s^{-2}(1+s^{-1})/\Delta\right]x_1(t_0) + \left[s^{-1}(1+s^{-1})/\Delta\right]x_3(t_0) + \left[s^{-2}(1+s^{-1})/\Delta\right]r(t_0)K_k$$

$$\Delta = (1+3s^{-1}+2s^{-2})$$

Comparing equations 4-1 with equations 2-4, it is noted that the coefficients of the state variables are different although the plant has the same transfer function. To solve equations 4-1, the final values of the state variables also differ from those used to solve equations 2-4. The final values for both sets of state variables are

Figure 2-4	Figure 4-3
$x_1(3T) = 1.0$	$x_1(3T) = 1.0$
$x_2(3T) = 0.0$	$x_2(3T) = 1.0$
$x_3(3T) = 0.0$	$x_3(3T) = 0.0$

The transfer functions of the digital controller, obtained using the digital program and a sampling period of T=1.0 seconds, are identical in both cases. This shows that the method of drawing the flow graph does not affect the results obtained for the digital controller.

4.3 FOURTH-ORDER SYSTEM WITH SIMPLE POLES

For the plant with transfer function $\frac{1}{s(s+1)(s+2)(s+3)}$ and a sampling period of T=1 second, the z-transform of the digital controller was found to be

$$D(z) = \frac{11.553 - 6.3886z^{-1} + 0.86460z^{-2} - 0.028636z^{-3}}{1.0000 + 0.83846z^{-1} + 0.19953z^{-2} + 0.0044535z^{-3}}$$

With the aid of the analog simulation flow graph of Figure 4-4, the transfer function was tested by the rep-op method and the results are shown in Figure 4-5.

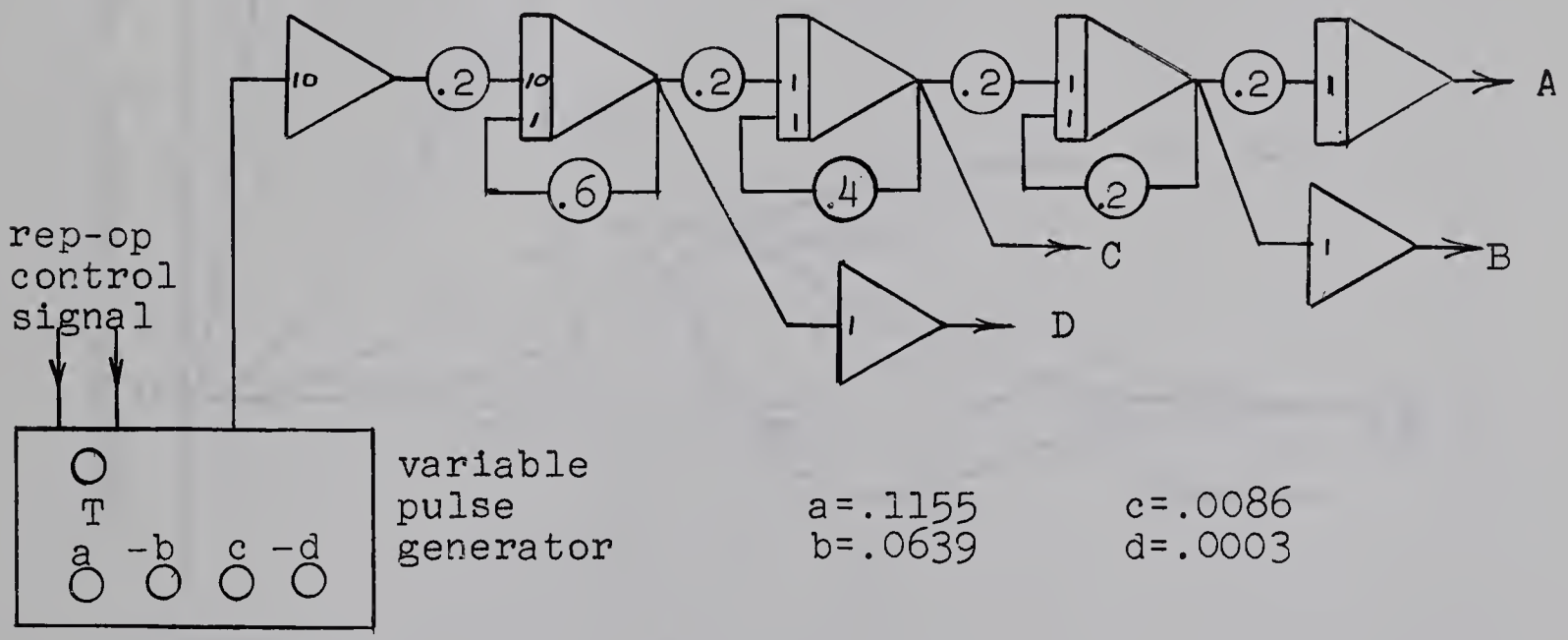
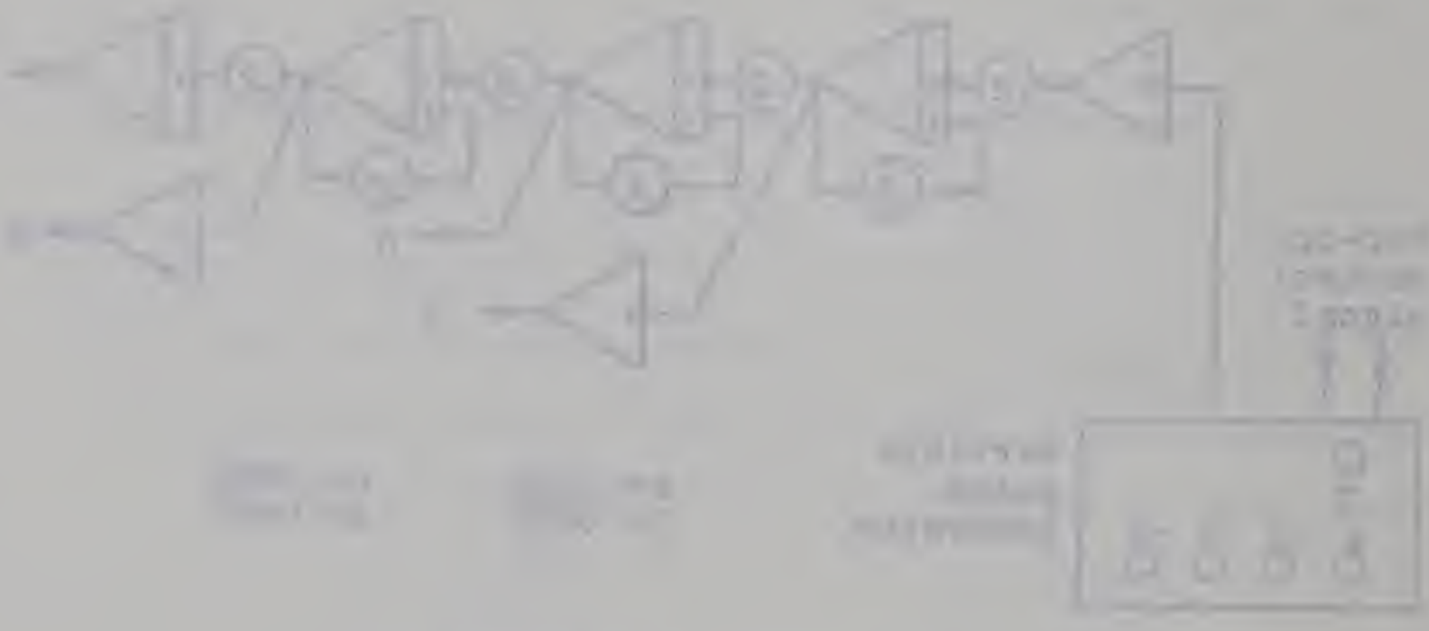


FIGURE 4-4. ANALOG SIMULATION FLOW GRAPH

4.4 THIRD-ORDER OSCILLATORY SYSTEM

The transfer function of a third-order oscillatory plant is of the form $\frac{1}{s(s^2+2\zeta\omega_n s+\omega_n^2)}$. The analog simulation flow graph of the plant for a damping ratio of $\zeta=\frac{1}{2}$, a natural frequency of $\omega_n=\pi$, and a period of oscillation = 2 seconds is shown in Figure 4-6. Transforms of the digital controllers were obtained using the digital program and tested



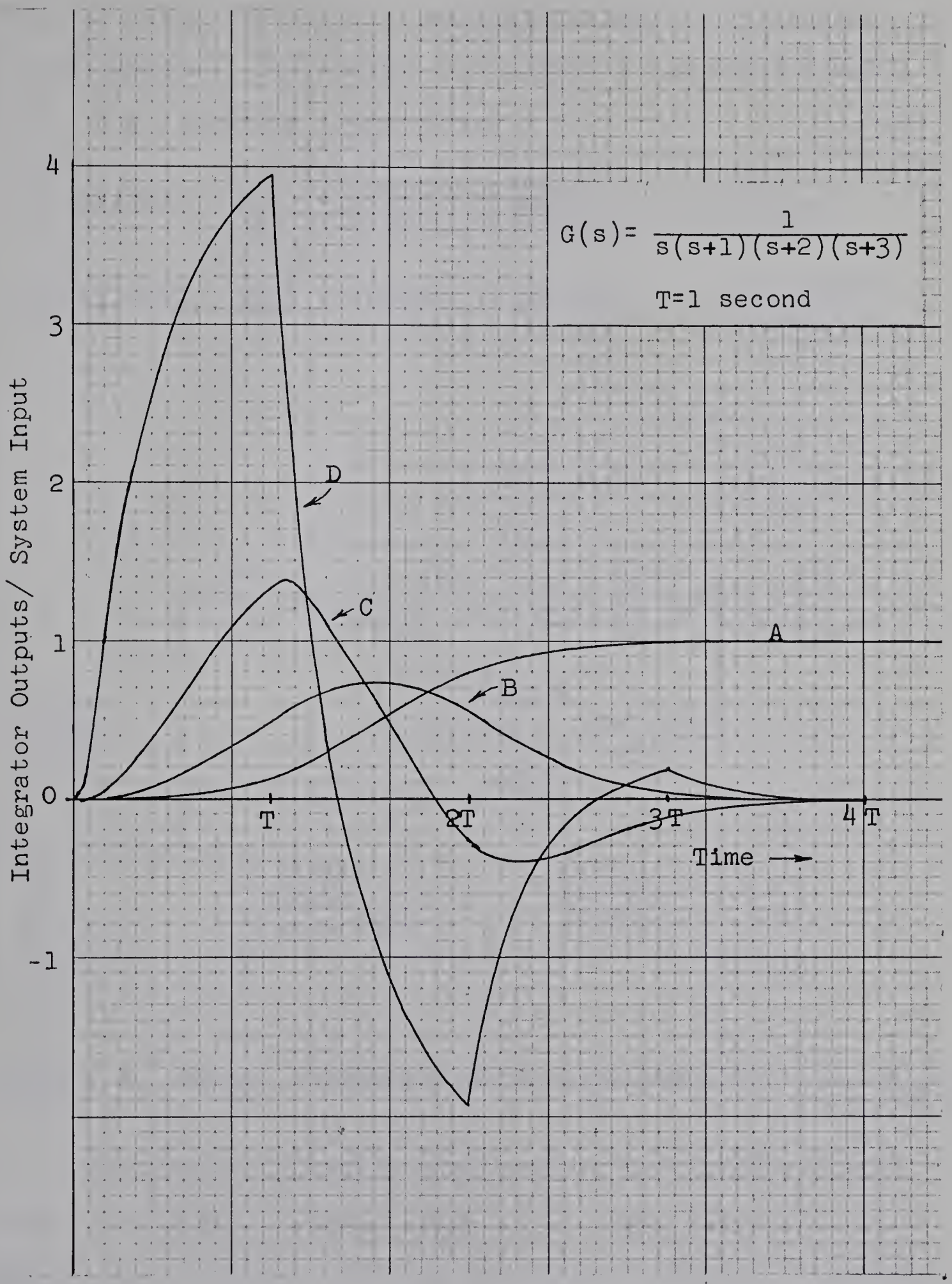


FIGURE 4-5. STATE VARIABLES OF FOURTH-ORDER SYSTEM

using the rep-op method for sampling periods of T=1 second and T=2 seconds. The results are shown in Figures 4-7 and 4-8 for the following z-transforms

$$\text{T=1 second:} \quad D(z) = \frac{6.9373 + 2.6325z^{-1} + 0.29979z^{-2}}{1.0000 + 0.57427z^{-1} + 0.071523z^{-2}}$$

$$\text{T=2 seconds:} \quad D(z) = \frac{5.2259 - 0.30087z^{-1} + 0.0097588z^{-2}}{1.0000 + 0.10156z^{-1} + 0.00057691z^{-2}}$$

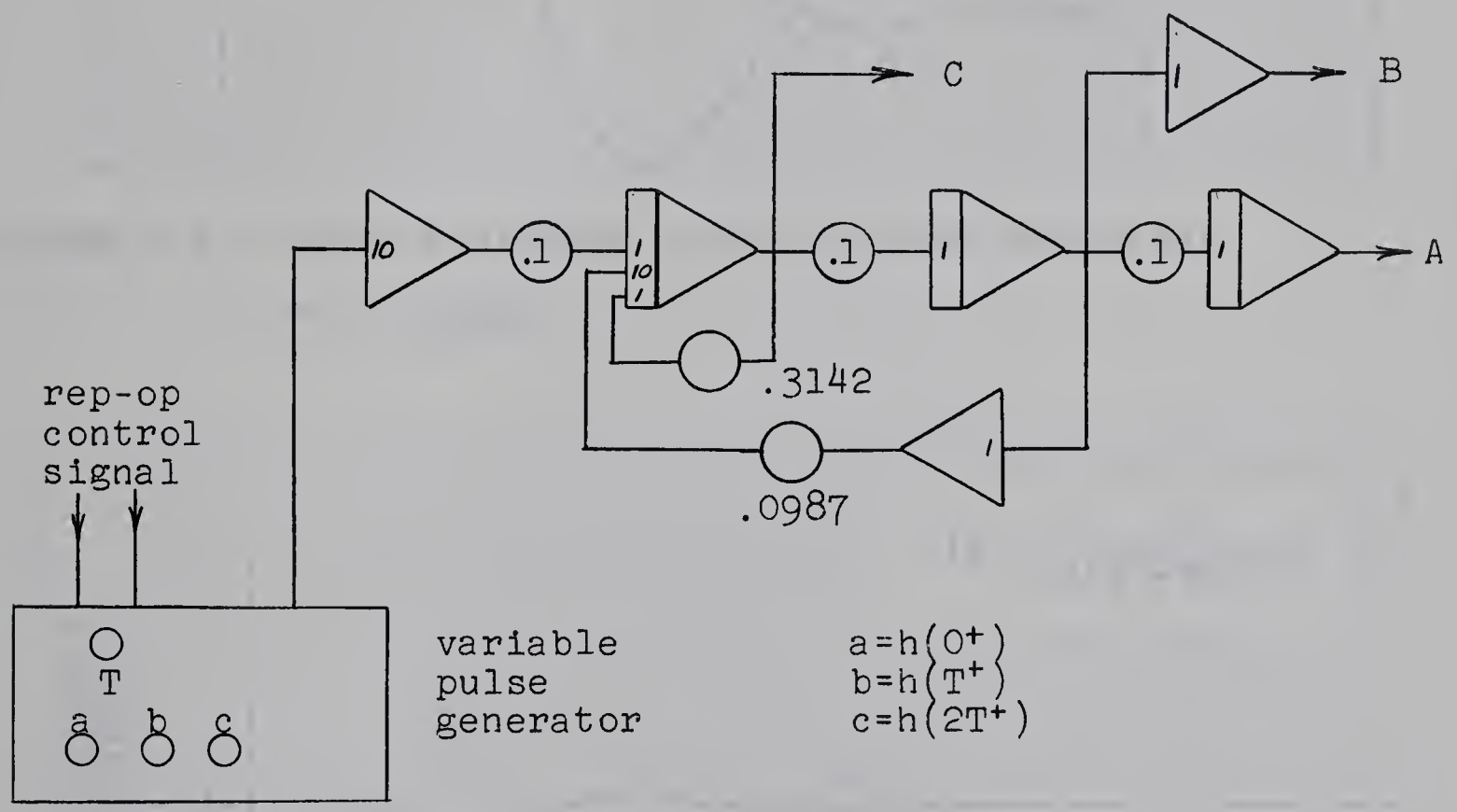
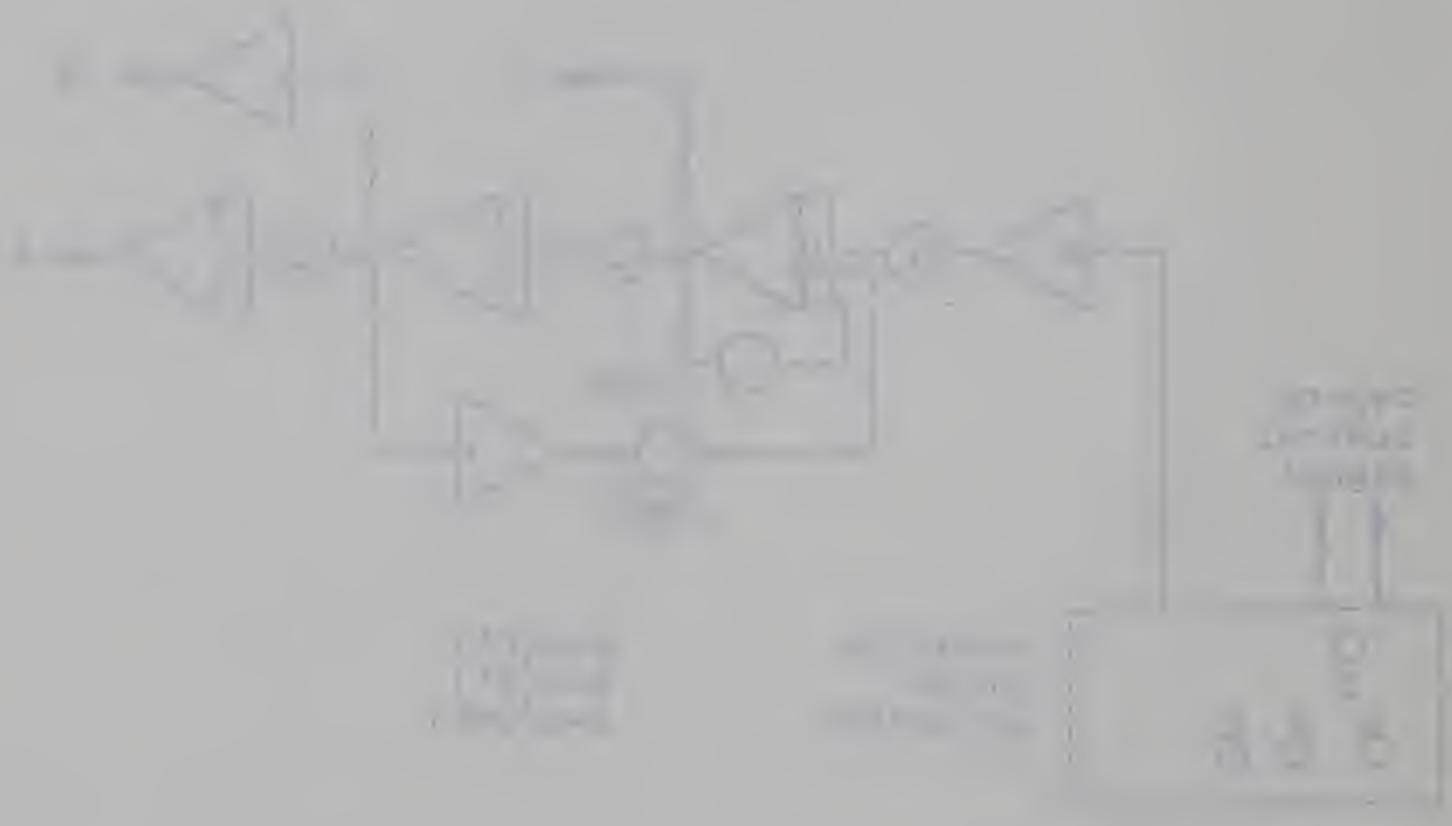


FIGURE 4-6. ANALOG SIMULATION FLOW GRAPH FOR $G(s) = \frac{1}{s(s^2+2\zeta\omega_n s+\omega_n^2)}$

Choosing a sampling period close to the time period of the oscillating system, results in slight overshoots as can be seen in Figure 4-8, but the output has almost reached the value of the input in one sampling period. The remaining two sampling periods are required to level off the output to



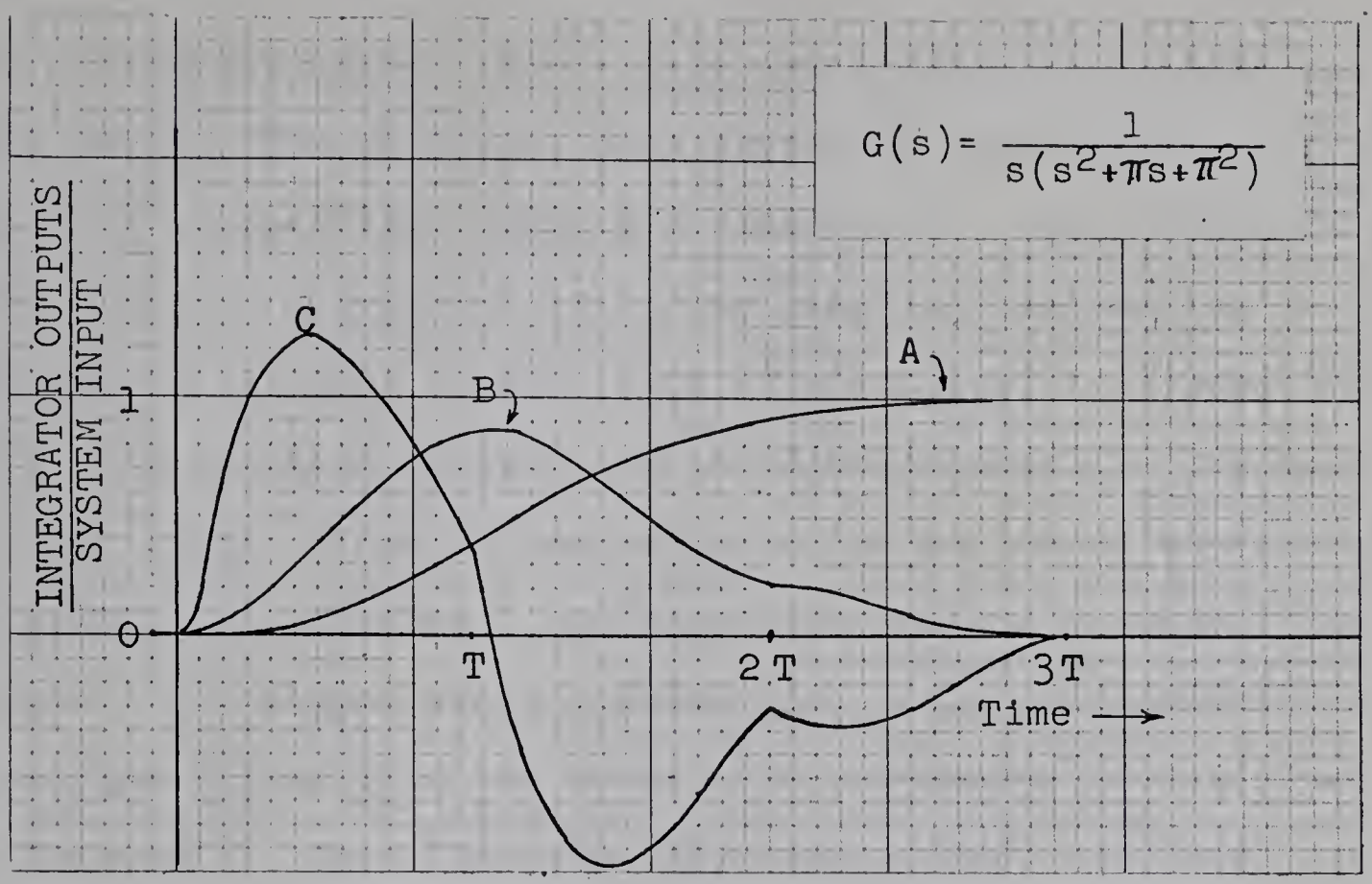


FIGURE 4-7. STATE VARIABLES OF OSCILLATORY SYSTEM FOR T= 1 SECOND

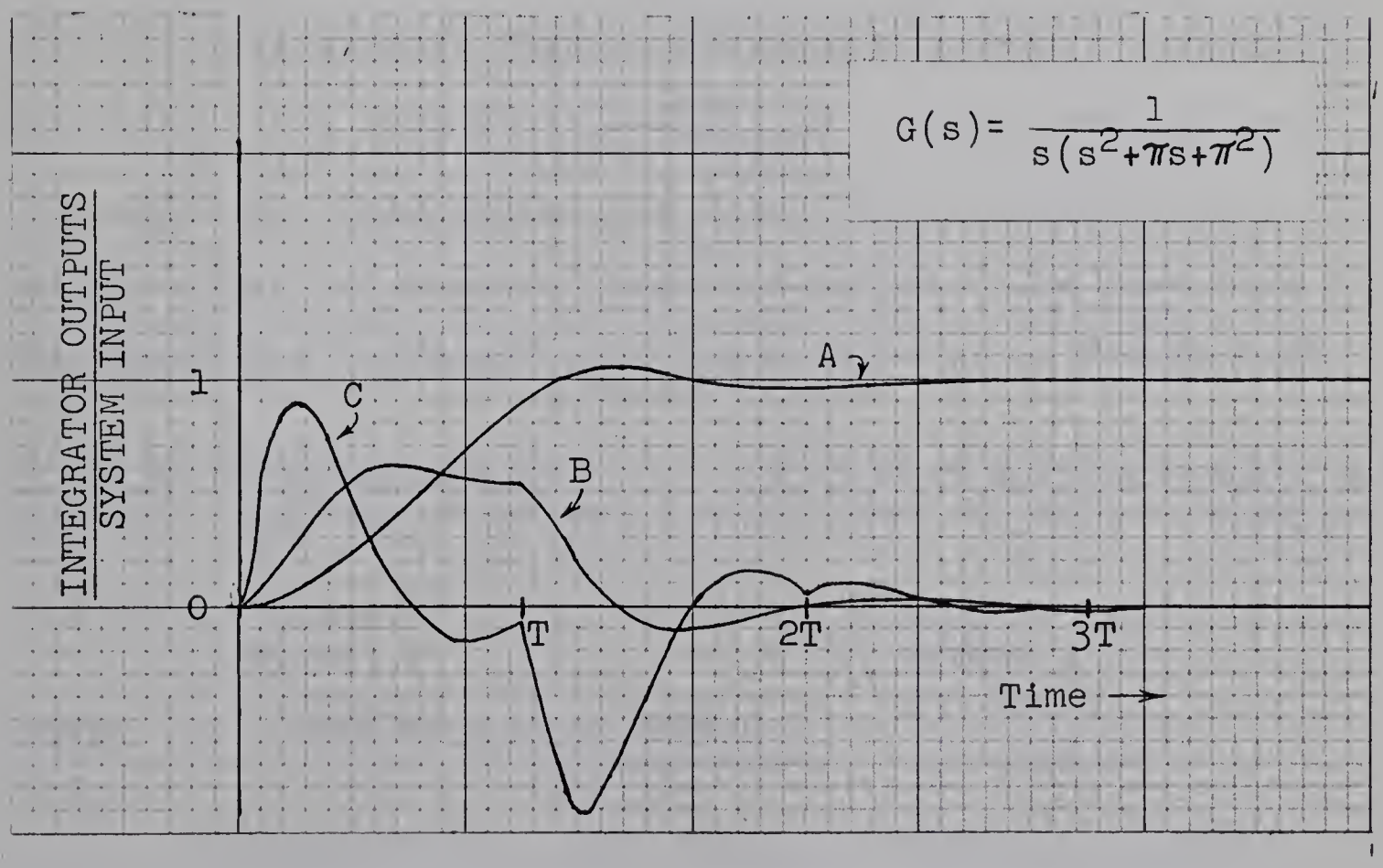
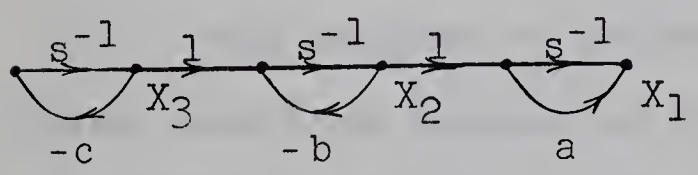


FIGURE 4-8. STATE VARIABLES OF OSCILLATORY SYSTEM FOR T= 2 SECONDS

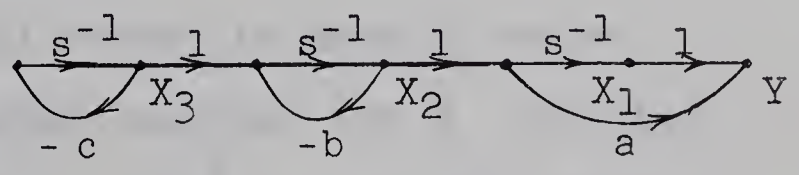
the designated value. For a sampling period of one-half the time period, the output is more stable as there is no tendency to oscillate about the final value. The value of the output in Figure 4-7 after two sampling periods is approximately equal to the value of the output in Figure 4-8 after one sampling period, but the time interval is the same in both cases. Thus, a knowledge of the period of oscillation is helpful in choosing a sampling rate that will control the output. For higher order systems with unfactored denominators, a program for solving the roots of a polynomial is described in Chapter V. Once the roots have been found, the period of oscillation is easily determined.

4.5 THIRD-ORDER SYSTEM WITH A ZERO

A third-order plant containing a zero has a transfer function of the form $G(s)=\frac{s+a}{s(s+b)(s+c)}$. When drawing the flow graph for this system, an extra node is required to add the effect of the zero onto the output. The open-loop flow graph may be drawn as in Figure 4-9a or in Figure 4-9b.



a. Method 1



b. Method 2

FIGURE 4-9. OPEN-LOOP FLOW GRAPHS

Applying Mason's gain formula to both flow graphs results in the same transfer function, but Method 1 cannot be set up directly on the analog computer as an additional ampli-

fier is required to add the "aX_2" term to the "X_1" term. Therefore, it is the value of Y and not X_1 that is subtracted from the input at the sampling instants. The complete flow graph, including a digital controller is shown in Figure 4-10.

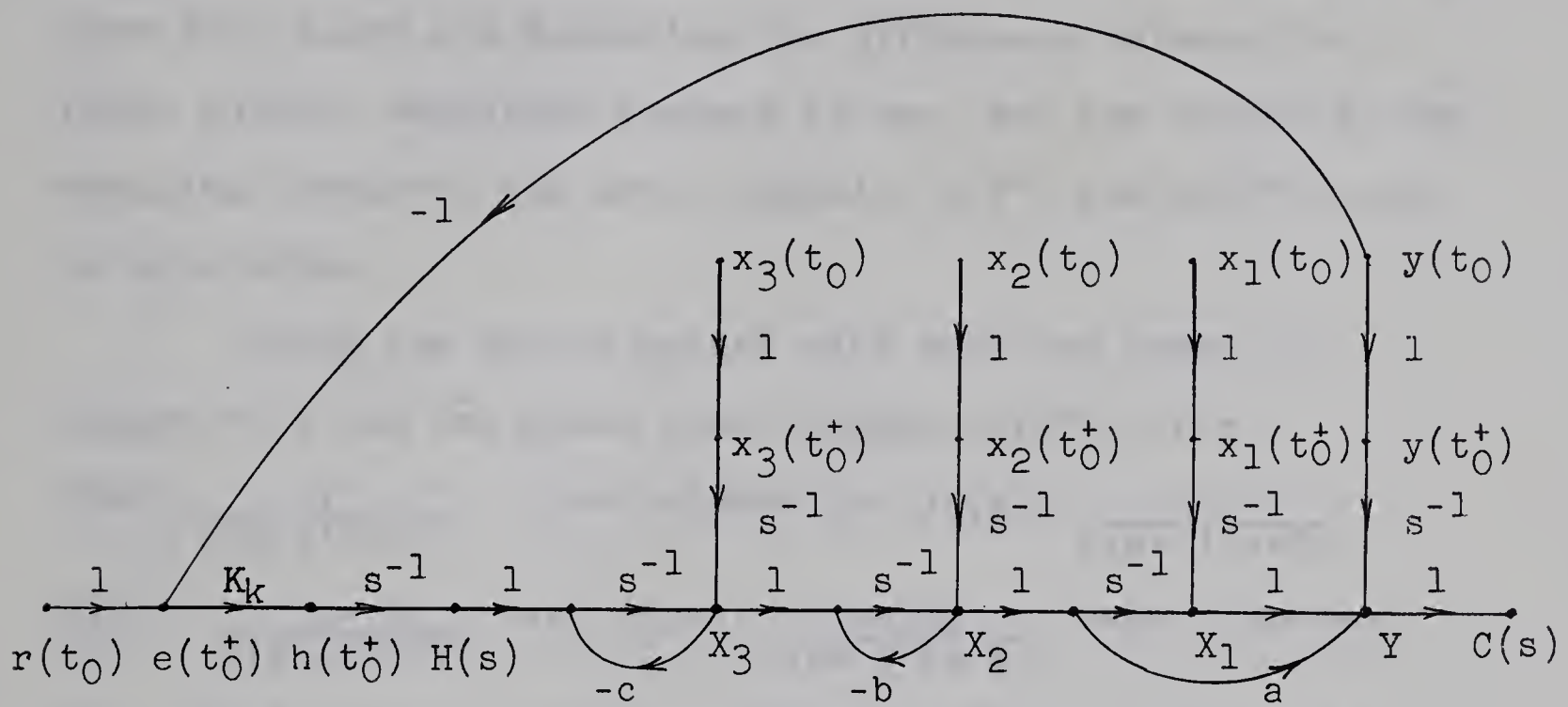


FIGURE 4-10. FLOW GRAPH FOR SYSTEM WITH A ZERO

To solve the system of Figure 4-10 requires four equations, one for each node, but the equation for Y is a linear combination of X_1 and X_2, thus there are three independent equations in three unknowns.

The program of Chapter III cannot be used to solve these equations because of the added equation for Y. Solving the equations for the plant $G(s)= \frac{s+.5}{s(s+1)(s+2)}$ and T=1 second, results in the $D(z)= \frac{3.6592 - 1.8413z^{-1} + 0.18218z^{-2}}{1.0000 + 0.32693z^{-1} - 0.065428z^{-2}}$

which has the same numerator as the D(z) obtained for $G(s)= \frac{1}{s(s+1)(s+2)}$, that is

$$D(z) = \frac{3.6592 - 1.8413z^{-1} + 0.18218z^{-2}}{1.0000 + 0.69246z^{-1} + 0.069043z^{-2}}$$

Thus if the zero term were neglected, the program of Chapter III could be used to solve for the inputs to the plant, $h(0^+)$, $h(T^+)$ and $h(2T^+)$. Applying these inputs to the open-loop plant and measuring the difference between the final output, magnitude assumed as one, and the output at the sampling instants, the error signals, $e(T^+)$ and $e(2T^+)$, can be calculated.

Using the rep-op method with the flow graph of Figure 4-11 and the above plant inputs, $h(nT^+)$, for $G(s)= \frac{1}{s(s+1)(s+2)}$, the outputs for $G_1(s)= \frac{1}{s(s+1)(s+2)}$, $G_2(s)= \frac{s+.5}{s(s+1)(s+2)}$ and $G_3(s)= \frac{s+1.5}{s(s+1)(s+2)}$ were observed. The graphs of the outputs are shown in Figure 4-12 for a sampling period of one second.

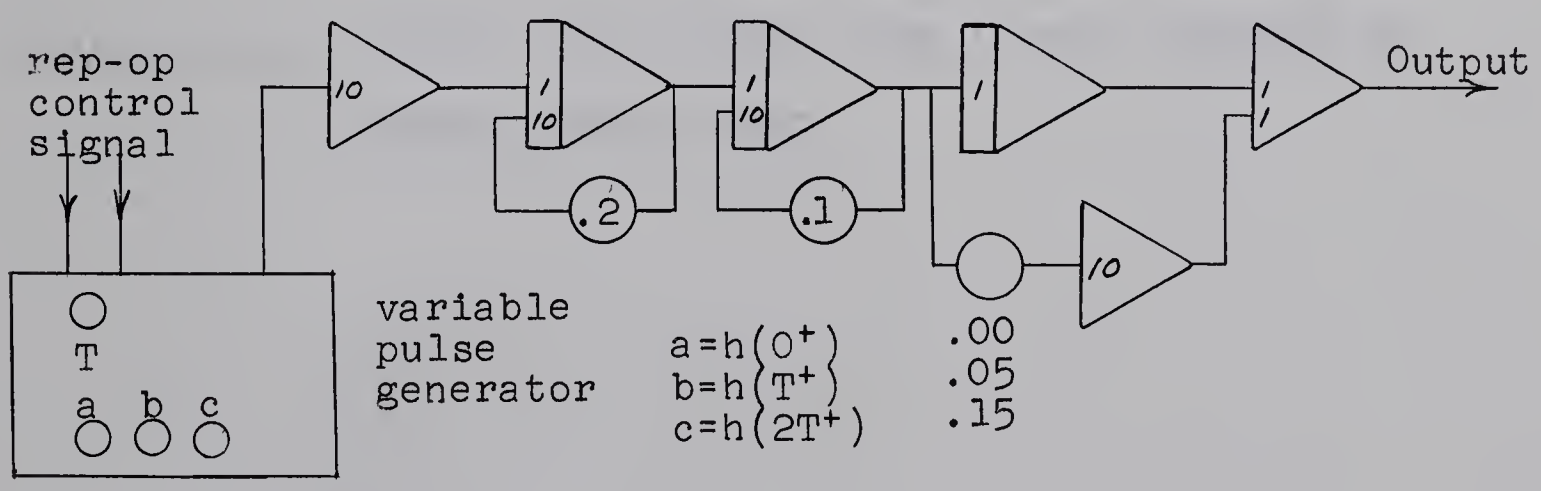
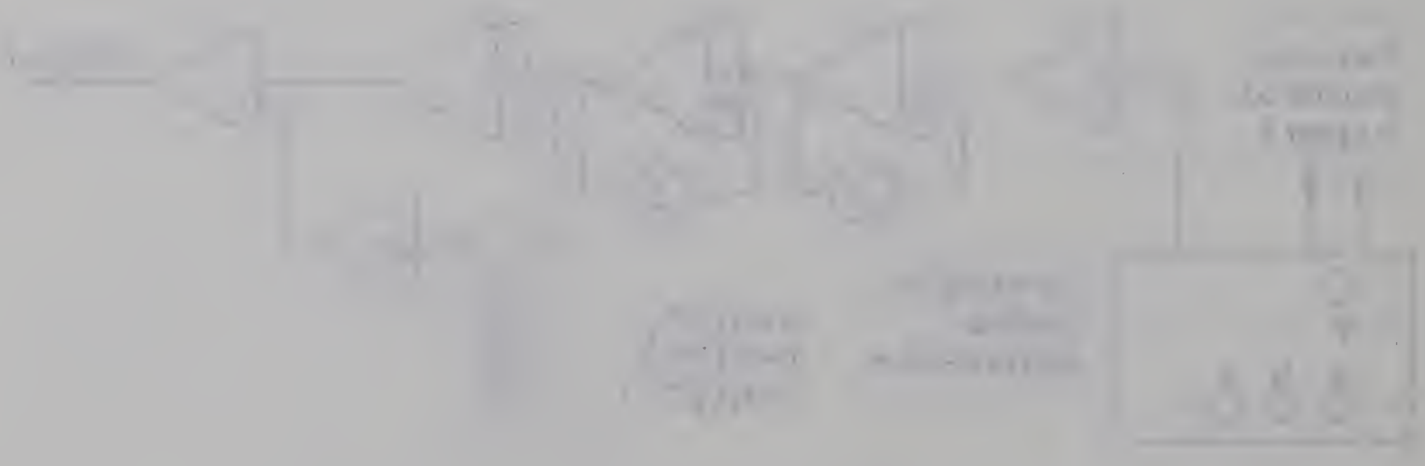


FIGURE 4-11. ANALOG FLOW GRAPH FOR SYSTEMS WITH A ZERO

All three outputs have become deadbeat after three sampling periods, but the further the zero is removed from



the origin in the s-plane, the larger the overshoot becomes. For systems with the pole and zero separated by a decade or more, no overshoot will occur in the output.

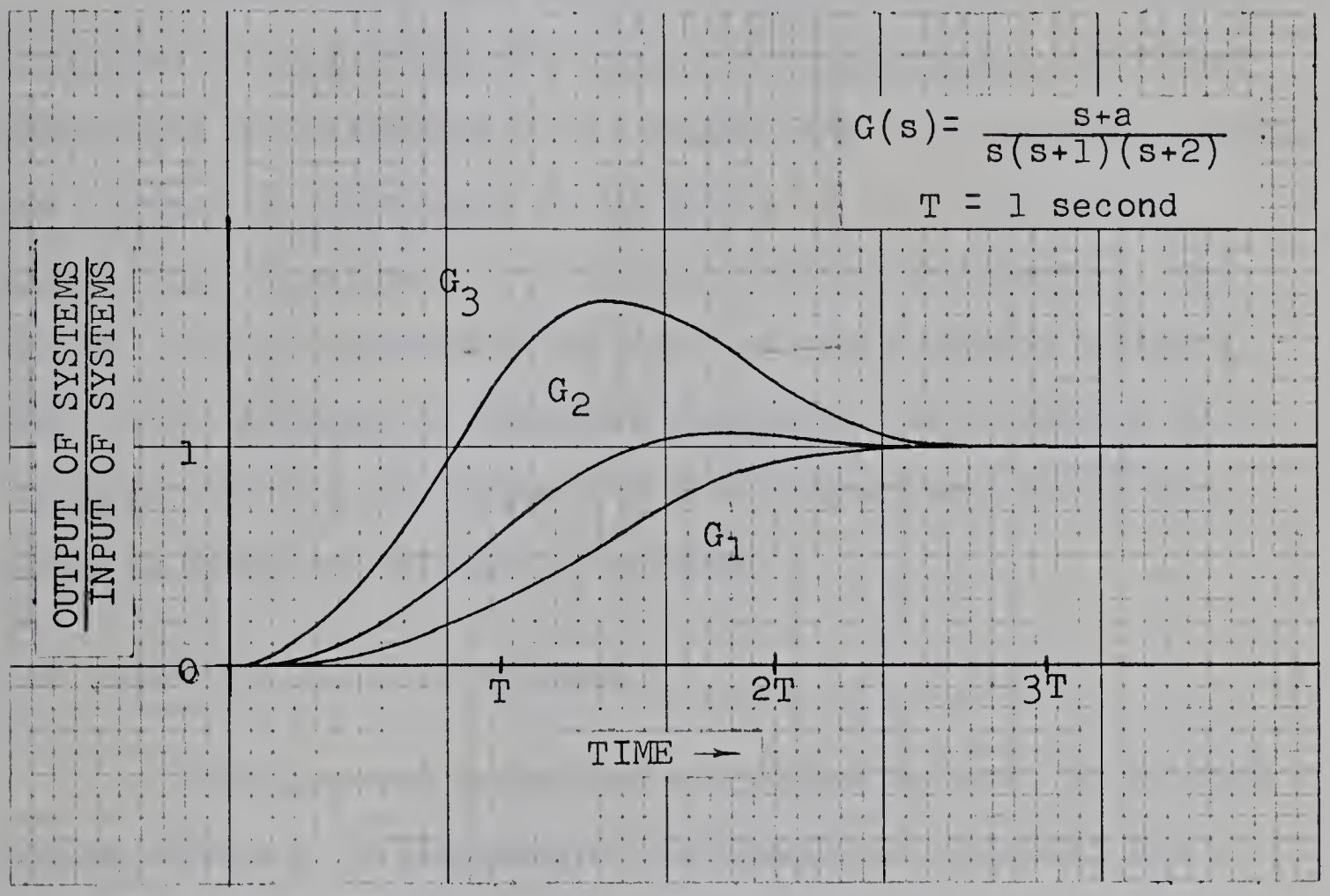


FIGURE 4-12. OUTPUTS FOR SYSTEMS WITH A ZERO COMPARED TO SYSTEM WITH NO ZERO

CHAPTER V

COMPUTATION OF ROOTS OF A POLYNOMIAL USING ROUTH'S STABILITY CRITERION

The concept and method of applying Routh's stability criterion to computing the roots of a polynomial was first introduced by Professor Y. J. Kingma and a subsequent program was written by Professor D. H. Kelly of the Electrical Engineering Department at the University of Alberta. The object of this thesis was to rewrite the original program, written in FORTRAN II computer language, into FORTRAN IV language, double precision the new program and check the accuracy obtained using the program.

5.1 SCOPE OF DIGITAL PROGRAM

The Routh's stability criterion is used in control system analysis to determine the number of roots of the characteristic equation of a closed-loop system that have positive real parts. This is accomplished by computing the Routh array and counting the number of sign changes in the first column of the array. Each sign change corresponds to a root with a positive real part. A description of Routh's stability criterion may be found in the literature[6].

The program finds the roots of a polynomial by successively shifting the imaginary axis and calculating the Routh array. The axis is shifted by increasingly smaller increments until it converges on the real part of the root to the accuracy specified in the input data. Roots of any order of multiplicity may be located using the program.

If there are no more than two complex pairs with the same real part, the imaginary parts of the root are then calculated. If there are three or more pairs of complex roots with the same real part, the coefficients of an auxilliary equation are printed out which may be inserted back into the program as new data to solve for the imaginary parts.

5.2 LIST OF TERMS USED IN PROGRAM

N ------ degree of polynomial

A(J) --- coefficients of polynomial

AA(J) -- coefficients of polynomial with shifted axis

C(I,J) - array of coefficients of $(s+1)$, $(s+1)^2$, ...$(s+1)^N$

B(I,J) - coefficients of Routh array

NR ----- number of roots still to be calculated

AL ----- distance axis is shifted from original position

NRR ---- number of roots with a real part located at AL

ALO ---- size of initial rightward step

ANO ---- size of initial leftward step (negative)

E1 ----- value used to test B(I,1) for zero

E2 ----- value that replaces B(I,1) if $B(I,1) \leq E1$

E3 ----- desired accuracy, equal to $\frac{\text{value of incremental shift}}{\text{value of real part of root}}$

5.3 CALCULATIONS PERFORMED USING COMPUTER PROGRAM

This program is dimensioned to accomodate polynomials of degree sixteen. For higher order polynomials, modify the dimension statement to

DIMENSION B(N2,N2), A(N1), AA(N2), C(N2,N2)

where N2=order of polynomial + 2

N1=order of polynomial + 1

Given a polynomial of the form

$$A(1)s^N + A(2)s^{N-1} + ... + A(N+1) \qquad .....(5-1)$$

the first step is to shift the imaginary axis of the s-plane to the left by an amount α, ANO, and calculate a new polynomial of the form

$$A(1)(s+\alpha)^N + A(2)(s+\alpha)^{N-1} + ... + A(N+1) \qquad .....(5-2)$$

which reduces to

$$AA(1)s^N + AA(2)s^{N-1} + ... + AA(N+1) \qquad .....(5-3)$$

The reduction of equation (5-2) to equation (5-3) is carried out with the aid of an array, C(I,J), containing the coefficients of $(s+1)$, $(s+1)^2$, ..., $(s+1)^N$ as shown in Figure 5-1.

	1	2	3	4	5	6	.....	N-1
1	1	1						
2	1	2	1					
3	1	3	3	1				
4	1	4	6	4	1			
5	1	5	10	10	5	1		
.								
.								
N	1	N	..	...	...	..	.....	1

FIGURE 5-1. C-ARRAY

The first row contains the coefficients of (s+1), the second row the coefficients of $(s+1)^2$, and so on down to the

N-th row which contains the coefficients of $(s+1)^N$.

The coefficients of equation (5-3) are formed thusly:

for N = 2, $AA(1) = A(1)$

$$AA(2) = A(1)C(2,2)_\alpha + A(2)$$

$$AA(3) = A(1)C(2,3)\alpha^2 + C(1,2)A(2)_\alpha + A(3)$$

for N = 3,

$$AA(1) = A(1)$$

$$AA(2) = A(1)C(3,2)_\alpha + A(2)$$

$$AA(3) = A(1)C(3,3)_\alpha{}^2 + A(2)C(2,2)_\alpha + A(3)$$

$$AA(4) = A(1)C(3,4)\alpha^3 + A(2)C(2,3)_\alpha{}^2 + A(3)C(1,2)_\alpha + A(4)$$

and for N = 4,

$$AA(1) = A(1)$$

$$AA(2) = A(1)C(4,2)_\alpha + A(2)$$

$$AA(3) = A(1)C(4,3)_\alpha{}^2 + A(2)C(3,2)_\alpha + A(3)$$

$$AA(4) = A(1)C(4,4)_\alpha{}^3 + A(2)C(3,3)_\alpha{}^2 + A(3)C(2,2)_\alpha + A(4)$$

$$AA(5) = A(1)C(4,5)\alpha^4 + A(2)C(3,4)_\alpha{}^3 + A(3)C(2,3)_\alpha{}^2 + A(4)C(1,2)_\alpha + A(5)$$

For higher values of N the coefficients are calculated in a similar manner.

The coefficients of equation (5-3) form the first two rows of the Routh array, B(I,J), as shown in Figure 5-2. The remainder of the Routh array is built up using equation (5-4).

$$B(I,J) = B(I-2,J+1) - B(I-1,J+1)*B(I-2,1)/B(I-1,1) \quad ...(5-4)$$

After each row is formed, the first element, B(I,1), is tested for a "zero" value. That is, is $B(I,1) < E1$, where E1 is a very small number? If it is less than E1, each remaining element in the row is tested for zeros, and if any of these have a value, B(I,1) is replaced by a small number E2.

	1	2	3	4
1	AA(1)	AA(3)	AA(5)	.....
2	AA(2)	AA(4)	AA(6)	.....
3	B(3,1)	B(3,2)	B(3,3)	.....
4	B(4,1)	B(4,2)	B(4,3)	.....
5	.....			

FIGURE 5-2. ROUTH ARRAY [B(I,J)]

If the entire row is zero, the coefficients are replaced by the coefficients of the derivative of the previous row.

Continuing this process of forming a row, testing it and making any necessary changes the complete Routh array is constructed. The sign changes in the first column are counted and thus the number of roots to the right of the shifted axis is determined. If this is equal to the order of the polynomial, a search is started, otherwise the axis is again shifted to the left and the procedure repeated until all roots are to the right of the axis.

The search is started by shifting the axis to the right by an amount ALO, and the calculations are carried out as above. If the number of roots to the right has not changed, the axis is again shifted to the right, otherwise an incremental shift to the left is made. Right and left shifts of decreasing size are made until the axis converges on the real part of the root to the desired accuracy E3.

The number of roots and AL, the real part of the root, are printed out. The number of roots found is subtracted

from the order of the polynomial and the result stored in a location NO. This is the value that is now used for a comparison when counting the number of sign changes in the first column of the Routh array. If the number of sign changes is equal to NO, a search is again started. The search is carried on until all the roots are found, that is when NO = 0.

If there are 2 to 5 roots with the same real part, there can be 2 or 4 complex roots which the program will calculate from the coefficients of the L-th row of the Routh array where

L = order of equation - number of roots

If there are 6 or more roots with the same real part, the coefficients of an equation whose roots are the imaginary components desired are printed out. The equation may be solved using the program and the coefficients as input data.

5.4 INPUT

The first data card contains the values of N, E1, E2, E3, ALO and ANO in their respective order. The field specification for punching the numerical values on the card is FORMAT (I5,5D10.2).

A choice of 0.01 for E2 can be used when solving any particular polynomial as this value is not critical. The values for E3, the desired accuracy, and E1, the value used in the zero test, are related in magnitude, E3 being the smaller. Although E3 is the desired accuracy, which when

achieved terminates the calculations, both E1 and E3 are used in tests designed to terminate the iterative calculations. If $DA/AL \leq E3$, where DA is the magnitude of the incremental step and AL the distance the root is from the original imaginary axis, then the specified accuracy has been obtained and the real part of the root, AL, is printed out. This is the dominant test for large roots, but if the root is very close to the original axis, that is, AL very small, then $DA \leq E1$ is the dominant test. This test ensures that roots less than 1 are correct to a specified number of decimal places.

For example, choosing $E3 = 10^{-10}$ and $E1 = 10^{-8}$ will ensure that roots with real parts less than one will be correct to eight places after the decimal. Also, roots with real parts less than 100 and larger than one will be correct to eight places after the decimal and to an accuracy of 10^{-8} per cent.

The values selected for ANO and ALO, the initial leftward and rightward axis shifts, have a bearing on the computation time. The choice for ANO should be approximately three times as large as that for ALO. As an example, for roots at s = -900, -1, ANO = -3, and ALO = 1, 300 iterations are required to reach the root at -900 and another 900 iterations to reach the root at -1. A choice of ANO = -30 and ALO = 10 reduces the iterations to 30 and 90 respectively, thus speeding up the solution. The saving in time is only a few seconds if the program is being run on the IBM 7040 computer, but is several minutes when using the IBM 1620

computer which is 60 times slower.

By studying the coefficients A(2) and A(N+1) of the given polynomial, the user of the program may obtain an approximation to the order of magnitude of the roots. The coefficient A(2) is the sum of the roots of the polynomial and A(N+1) is the product of the roots.

The remaining data cards contain the coefficients of the polynomial. They are punched onto the cards in the order A(1), A(2), ..., A(N+1) according to the field specification FORMAT (4D18.10).

5.5 FORTRAN IV COMPUTER PROGRAM FOR ROOT SOLUTION

The flow chart for this program is in Appendix B. The program is interspersed with COMMENT statements which indicate the calculations that are to follow.

```
$IBFTC RCM
      DOUBLE PRECISION B,A,AA,C,E1,E2,E3,ALO,ANO,DA,H,AL,R1,
     1R2,R3,R4
      DIMENSION B(18,18),A(17),AA(18),C(18,18)
    1 READ (5,80) N,E1,E2,E3,ALO,ANO
   80 FORMAT (I5,5D10.2)
      N1=N+1
      N2=N+2
      AL=ANO
      KM1=0
      NN=0
      NR=0
```

```
      KK=0
      KS=0
      DA=ALO/2.DO
C  ZERO OUT AA ARRAY
      DO 2 I=1,N2
    2 AA(I)=0.DO
C  CALCULATE COEFFICIENTS OF (S+1), (S+1)**2, ....(S+1)**N
      C(1,1)=1.DO
      C(1,2)=1.DO
      DO 3 I=2,N
      C(I,1)=1.DO
      C(I,I+1)=1.DO
      DO 3 J=2,I
    3 C(I,J)=C(I-1,J)-C(I-1,J-1)
      PE=E3*100.DO
      WRITE (6,81) N,PE
   81 FORMAT(22H1 ORDER OF POLYNOMIAL=,I3/16H PERCENT ERROR=
     1D10.2/)
      READ (5,82) (A(I),I=1,N1)
   82 FORMAT (4D18.10)
      WRITE (6,83) (A(I), I=1,N1)
   83 FORMAT (1X,4D17.10)
C  ZERO OUT ROUTH ARRAY
      AA(1)=A(1)
      DO 5 I=1,N1,2
      JJ=N2/2-I/2+1
      DO 5 J=1,JJ
      B(I,J)=0.DO
```

```
    5 B(I+1,J)=0.D0
      GO TO 44
C   FILL IN FIRST 2 ROWS OF ROUTH ARRAY
   40 NN=0
      NR=0
      DO 6 I=1,N1,2
      J=(I+2)/2
      B(1,J)=AA(I)
    6 B(2,J)=AA(I+1)
C   TEST SECOND ROW FOR ZERO COEFFICIENTS
      K=2
      JJ=N2/2
      KM1=K-1
      NN=N2-K
      IF(DABS(B(K,1)).GE.E1) GO TO 19
   10 DO 11 J=2,JJ
      IF(DABS(B(K,J)).GE.E1) GO TO 13
   11 CONTINUE
      DO 12 J=1,JJ
      H=NN+2-2*J
   12 B(K,J)=B(KM1,J)*H
      GO TO 19
   13 B(K,1)=E2
   19 IF(B(K,1)*B(KM1,1).GE.0.) GO TO 15
   14 NR=NR+1
C   FILL IN NEXT ROW
   15 DO 7 I=3,N1,2
```

```
      JJ=N2/2-I/2
      DO 8 J=1,JJ
      B(I,J)=B(I-2,J+1)-B(I-2,1)*B(I-1,J+1)/B(I-1,1)
    8 CONTINUE
C   TEST FOR ZERO COEFFICIENTS
      K=I
      KM1=K-1
      NN=N2-K
      IF(DABS(B(K,1)).GE.E1) GO TO 79
   70 DO 71 J=2,JJ
      IF(DABS(B(K,J)).GE.E1) GO TO 73
   71 CONTINUE
      DO 72 J=1,JJ
      H=NN+2-2*J
   72 B(K,J)=B(KM1,J)*H
      GO TO 79
   73 B(K,1)=E2
C   TEST FOR SIGN CHANGE
   79 IF(B(K,1)*B(KM1,1).GE.0.) GO TO 75
   74 NR=NR+1
C   FILL IN NEXT ROW
   75 DO 9 J=1,JJ
      B(I+1,J)=B(I-1,J+1)-B(I-1,1)*B(I,J+1)/B(I,1)
    9 CONTINUE
C   TEST FOR ZERO COEFFICIENTS
      K=I+1
      KM1=K-1
```

```
      NN=N2-K
      IF(DABS(B(K,1)).GE.E1) GO TO 99
   90 DO 91 J=2,JJ
      IF(DABS(B(K,J)).GE.E1) GO TO 93
   91 CONTINUE
      DO 92 J=1,JJ
      H=NN+2-2*J
   92 B(K,J)=B(KM1,J)*H
      GO TO 99
   93 B(K,1)=E2
C  TEST FOR SIGN CHANGE
   99 IF(B(K,1)*B(KM1,1).GE.0) GO TO 7
   94 NR=NR+1
    7 CONTINUE
C  IF ALL ROOTS ARE TO THE RIGHT AND CONVERGENCE HAS NOT
C  STARTED, SHIFT AXIS LEFT BY ANO
      IF(N.LE.NR) GO TO 24
   21 IF (KS.GT.0) GO TO 26
   22 AL=AL+ANO
      GO TO 44
   24 IF(KS.NE.0) GO TO 26
   25 NO=NR
   26 IF(NO.LE.0) GO TO 1
   31 IF(NO.GT.NR) GO TO 34
   33 IF(KK.NE.0) GO TO 35
   37 AL=AL+ALO
      DA=ALO/2.D0
```

```
      GO TO 38
   35 AL=AL+DA
   41 DA=DA/2.DO
      GO TO 38
   34 AL=AL-DA
      KK=1
      IF(DABS(DA/AL).LE.E3) GO TO 42
   43 IF(DABS(DA).GT.E1) GO TO 41
      IF(DABS(DA).LE.E1) GO TO 42
   38 KS=1
C  CALCULATE AA(2).....AA(N+1)
   44 DO 30 I=2,N1
      AA(I)=A(I)
      N2I=N2-I
      IM1=I-1
      DO 30 J=1,IM1
      IMJ=I-J
      N1J=N1-J
   30 AA(I)=AA(I)+C(N1J,N2I)*A(J)*(AL**IMJ)
      GO TO 40
   42 NRR=NO-NR
      KK=0
      WRITE (6,84)NRR,AL
   84 FORMAT (1HO,I3,23H ROOTS WITH REAL PART= D25.16/)
      L=N1-NRR
      LP=(L+2)/2
C  TEST FOR 0,2,4 OR MORE IMAGINARY PARTS
```

```
      IF(NRR.LE.1) GO TO 25
   51 IF(NRR.LE.3) GO TO 55
   52 IF(NRR.LE.5) GO TO 53
   56 L2=L/2
      WRITE (6,57)
   57 FORMAT (48H THE IMAG PARTS ARE THE SQRTS OF THE ROOTS
     1OF AN/)
      WRITE(6,58)L2
   58 FORMAT (42H AUXILIARY EQTN A(1)S**N...A(N+1) OF ORDER,
     1I3/)
      WRITE (6,59) (B(L,M),M=1,LP
   59 FORMAT (41H WHOSE COEFF IN DESCENDING ORDER OF S ARE,/
     11X,6D18.10/)
      IF(L.LE.2*L2) GO TO 25
   61 WRITE (6,62)
   62 FORMAT (26H A SINGLE ROOT ALSO EXISTS/)
      GO TO 25
   55 R1=DSQRT(DABS(B(L,2)/B(L,1)))
      R2=-R1
      WRITE (6,85) R1,R2
   85 FORMAT (33H 2 OF THESE HAVE COMPLEX PARTS AT/2D25.16/)
      GO TO 25
   53 R1=-B(L,2)/(2.D0*B(L,1))
      R3=DSQRT(R1*R1-B(L,3)/B(L,1))
      R4=DSQRT(DABS(R1+R3))
      R2=DSQRT(DABS(R1-R3))
      R1=-R2
      R3=-R4
```

```
      WRITE (6,86) R2,R1,R4,R3
   86 FORMAT (33H 4 OF THESE HAVE COMPLEX PARTS AT/4D25.16/)
      GO TO 25
      END
```

CHAPTER VI

VALIDATION OF RESULTS OBTAINED USING ROUTH'S CRITERION FOR POLYNOMIAL ROOT SOLUTION

This chapter compares the results obtained using the computer program of Chapter V with those obtained using a program from the SHARE Library of the Computing Center at the University of Alberta.

6.1 SHARE LIBRARY PROGRAM C2-3332

This program can be used to find the roots of a given polynomial of any order or to generate a polynomial from the given roots. When solving for the roots of a polynomial, the program generates a polynomial from the solved roots which can be compared to the given polynomial. This serves as a built-in accuracy check of the calculated roots.

The roots are found using a combination of Newton and Muller approximations. Muller iterations provide a first approximation to the root and the Newton iterations continue the solution until the root is found to the desired accuracy. The root, roots if complex conjugates, is extracted from the polynomial and the solution continues with the reduced polynomial.

All calculations are carried out using double precision arithmetic, but the solutions are printed out in single precision arithmetic.

6.2 RESULTS OBTAINED USING ROUTH'S CRITERION PROGRAM

The SHARE Library program C2-3332 was used as a

standard to check the accuracy of the program based on Routh's criterion.

An ill-conditioned polynomial was chosen to test the accuracy of the programs, as it is one of the most difficult types of polynomials to solve. Small changes in the coefficients of an ill-conditioned polynomial will cause large changes in the values of the roots. The ill-conditioned polynomial shown below was chosen from the article "The Evaluation of the Zeros of Ill-Conditioned Polynomials" by J. H. Wilkinson printed in Numerische Mathematik, Part 1, Springer-Verlag, 1959.

$$2.03253121s^{16} + 3.4356048s^{15} + 25.1783048s^{14} + 37.651096s^{13} + 128.218748s^{12} + 166.44768s^{11} + 345.07256s^{10} + 378.908s^{9} + 524.327s^{8} + 468.88s^{7} + 443.576s^{6} + 304.08s^{5} + 190.68s^{4} + 89.6s^{3} + 32.8s^{2} + 8s + 1 = 0 \quad \text{.....(6-1)}$$

The solutions given in the literature were correct to 15 decimal places, but were rounded-off to 9 decimals as shown in Table 6.1. Solutions of the polynomial using the SHARE program and the Routh's criterion program are also shown in Table 6.1. The values used in the Routh's criterion program for E1 and E3 were $1x10^{-8}$ and $1x10^{-10}$ respectively.

Table 6.1 shows that the roots obtained using the SHARE program are in agreement with the correct values for the roots to eight significant figures. The results obtained using the Routh program were in agreement to nine significant figures with the correct values, except for roots 5 and 6 which only agreed to eight significant figures, but only 14

ROOTS	CORRECT VALUE OF ROOT	RESULTS USING SHARE PROGRAM C2-3332	RESULTS USING ROUTH CRITERION PROGRAM
1,2	-0.293504529 ±j0.143499296	-0.29350453 ±j0.14349930	-0.293504529 ±j0.143499296
3,4	-0.224470057 ±j0.450927958	-0.22447006 ±j0.45092796	-0.224470057 ±j0.450927958
5,6	-0.147623780 ±j0.771757201	-0.14762378 ±j0.77175720	-0.147623780 ±j0.771757200
7,8	-0.0900399887 ±j1.06119206	-0.090039989 ±j1.0611921	-0.0900399887 ±j1.06119206
9,10	-0.0508644356 ±j1.29691128	-0.050864436 ±j1.2969113	-0.0508644356 ±j1.29691128
11,12	-0.0256687105 ±j1.47437714	-0.025668711 ±j1.4743771	-0.0256687105 ±j1.47437714
13,14	-0.0104935501 ±j1.59629550	-0.010493550 ±j1.5962955	-0.0104935501 ±j1.59629550
15,16	-0.00248920244 ±j1.66712036	-0.0024892024 ±j1.6671204	PROGRAM TERMINATED

TABLE 6.1 SOLUTIONS FOR ILL-CONDITIONED POLYNOMIAL (6-1)

of the 16 roots were calculated. An excessive number of floating point traps were encountered when searching for roots

15 and 16 and the calculations were thus terminated. Floating point traps resulting in the output of underflow messages were also encountered when the computer was searching for roots 11, 12, 13 and 14, but there were not enough of them to terminate the calculations.

To further compare the results of the two programs, a second polynomial shown below, not ill-conditioned, was solved and the roots obtained listed in Table 6.2.

$$s^9 + 21.077365s^8 + 173.21313s^7 + 758.54868s^6 + 2185.2366s^5 + 4804.673s^4 + 7758.6178s^3 + 8765.4409s^2 + 7417.3856s + 1692.535 = 0 \quad .....(6\text{-}2)$$

ROOTS	RESULTS USING SHARE PROGRAM C2-3332	RESULTS USING ROUTH CRITERION PROGRAM
1	-7.7770889	-7.7770889
2	-5.4321411	-5.4321411
3,4	-3.2139902 ±j0.12334476	-3.2139902 ±j0.12334476
5,6	-0.49999812 ±j1.7321397	-0.49999812 ±j1.7321397
7	-0.32167519	-0.32167519
8,9	-0.059241578 ±j1.9236846	-0.059241578 ±j1.9236846

TABLE 6.2 SOLUTIONS FOR POLYNOMIAL (6-2)

The values of the roots obtained using Routh's criterion were rounded-off to eight significant figures.

An examination of Table 6.2 shows that the results obtained in both cases are identical. Although all the roots were calculated using the Routh's criterion program, floating point traps were encountered while the computer was searching for root number 7.

The reason for the floating point traps is the extremely small numbers calculated when determining the coefficients AA(I) of the polynomial with shifted axis. These occur when the shifted axis is close to the original axis, that is AL very small.

For the case of the ill-conditioned polynomial, when the computer was searching for the roots with real part at -0.00248920244, the axis would be between -0.001 and -0.003 sometime during the search. Raising any number in this range to the sixteenth power, produces a number less than $1x10^{-38}$ which will interrupt the computer. The floating point trap substitutes a zero and the calculations proceed, but the amount of axis shift, AL, eventually becomes small enough to cause another underflow and thus the solution is terminated before completion.

In the case of the polynomial (6-2), the underflows were also caused by numbers less than $1x10^{-38}$. They arose when the axis was shifted 0.5 units to the right, ALO=0.5, from -0.49999812 thus setting AL at 0.00000188. Raising this number to the ninth power caused the underflow. But in this incidence the floating point trap substituted zero and the

calculations proceeded until the solution was completed.

6.3 TIME REQUIRED FOR CALCULATIONS

The time required for calculations, that is the actual program running time or object time, and the total time required to process the program are shown in Table 6.3 for both the Routh program and the SHARE program.

EQUATION	ROUTH PROGRAM		SHARE PROGRAM	
	OBJECT TIME	TOTAL TIME	OBJECT TIME	TOTAL TIME
6-1	27	83	11	84
6-2	22	58	4	52

TABLE 6.3 TIME (IN SECONDS) REQUIRED FOR CALCULATIONS

Although the object time of the SHARE program is much less than that of the Routh program for the polynomial solutions considered, the total times required to run the programs are approximately the same in both cases. Thus, if time was a factor in choosing a program for a particular solution, the SHARE program offers the advantage of a shorter computer running time.

6.4 CONCLUSIONS

The program based on Routh's stability criterion will

solve for the roots of almost any polynomial to the accuracy stipulated in the input data. Polynomials for which all the roots may not be calculated are those which do not satisfy the conditions $(AL)^N \geq 10^{-38}$ and $(AL)^N \leq 10^{38}$ where AL is the real part of the root and N is the degree of the polynomial. Failure of the roots to satisfy these conditions causes an excessive number of underflows or overflows respectively and the calculations are terminated in either case.

In engineering practice, polynomials with roots not satisfying the above conditions are not normally encountered. Thus the engineer can use this program with confidence that the solutions will be correct and complete. If by chance any roots did lie outside these extremes, the roots calculated and printed out would be correct, but the solution would not be complete.

This program may appeal to the electrical engineer because it is based on Routh's stability criterion with which he is familiar. This allows him to vary the input paramaters with an understanding of the effects the changes will have on the calculations.

BIBLIOGRAPHY

(1) ADAMS, W.S., FORTRAN IV: A Programming Language, University of Alberta, 1964.

(2) D'AZZO, JOHN J., HOUPIS, CONSTANTINE H., Feedback Control System Analysis and Synthesis, McGraw-Hill, 1960.

(3) KINGMA, Y.J., Signal Flow Graphs, University of Alberta, 1965.

(4) KUO, BENJAMIN C., Analysis and Synthesis of Sampled-Data Control Systems, Prentice-Hall, 1963.

(5) McCRACKEN, DANIEL D., DORN, WILLIAM S., Numerical Methods and FORTRAN Programming, Wiley, 1966.

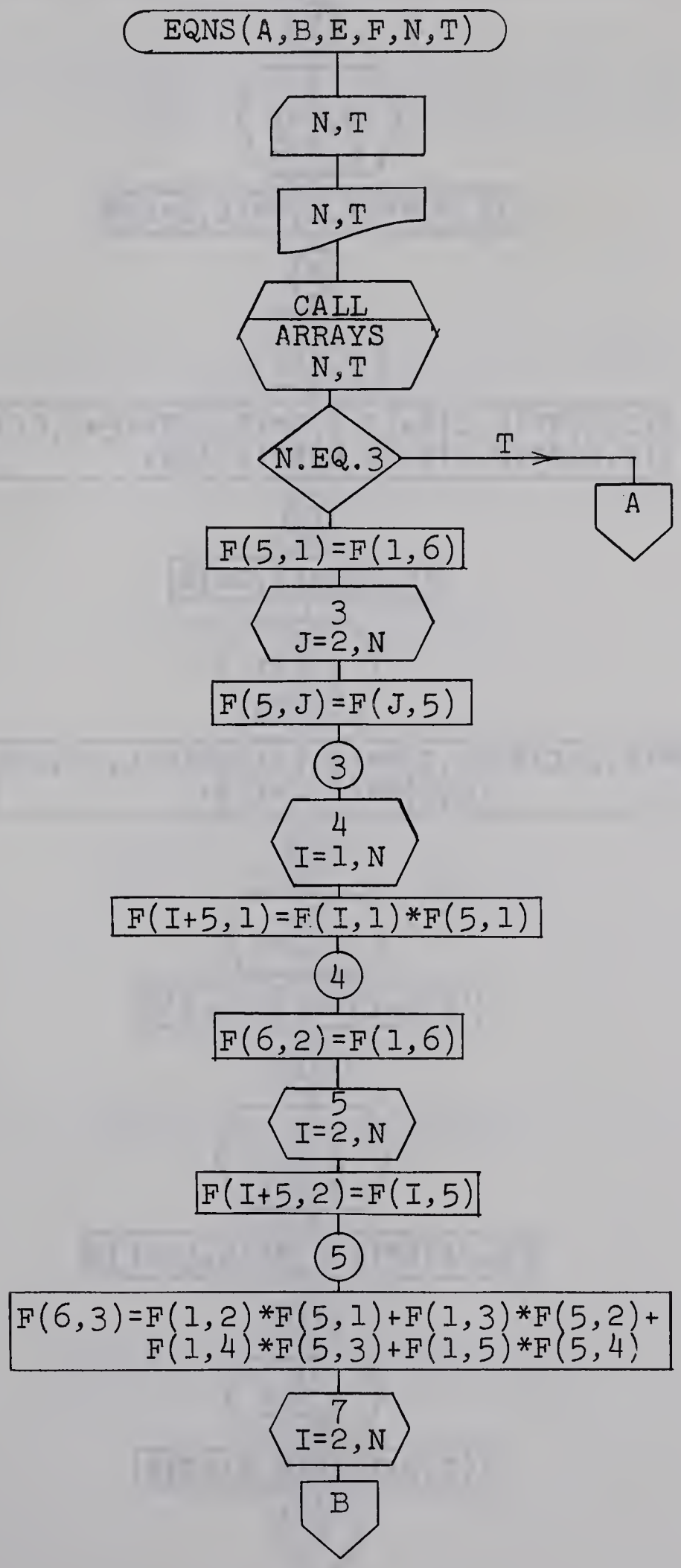
(6) RAGAZZINI, J.R., FRANKLIN, G.F., Sampled-Data Control Systems, McGraw-Hill, 1958.

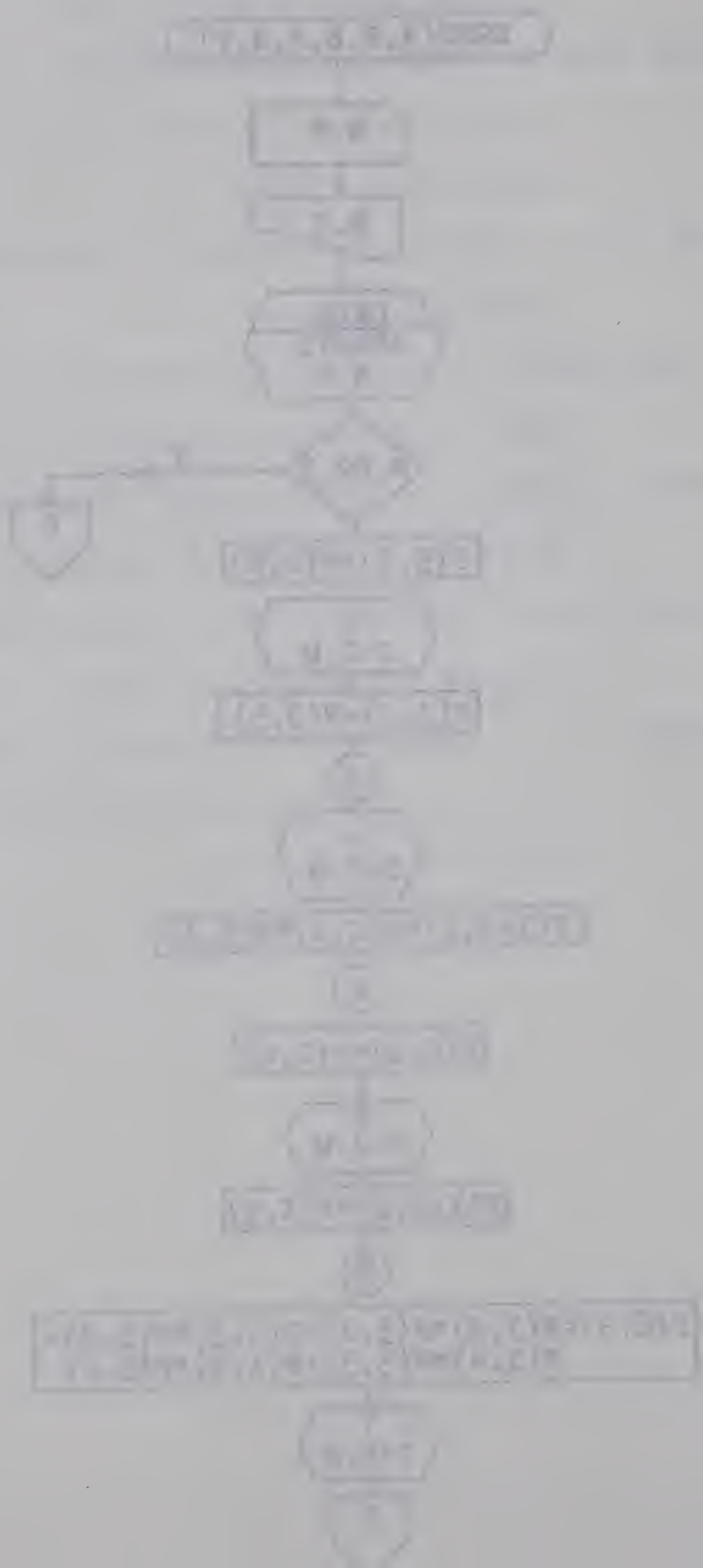
(7) WILKINSON, J.H., Numerische Mathematik, The Evaluation of the Zeros of Ill-Conditioned Polynomials, Springer-Verlag, 1959.

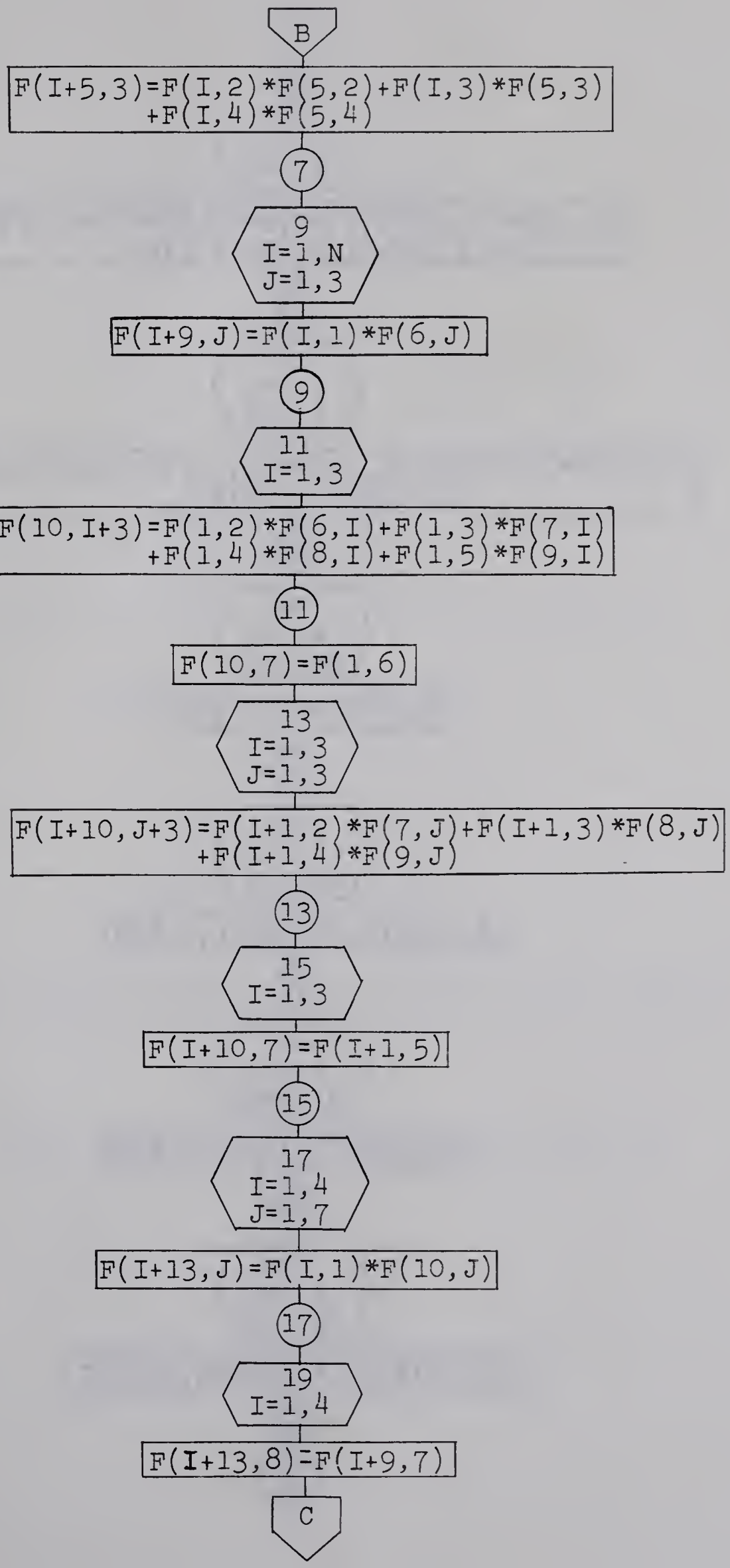
APPENDIX A

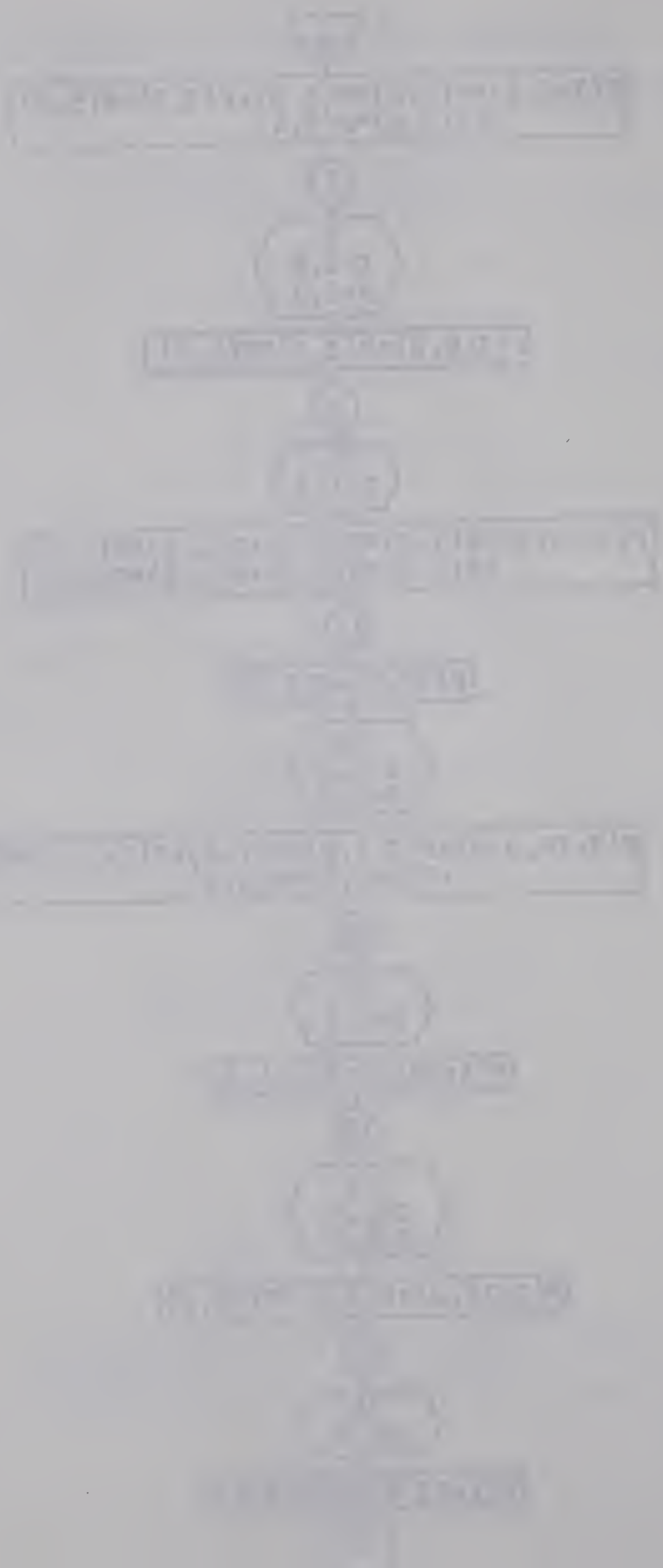
FLOW CHART FOR PROGRAM OF CHAPTER III

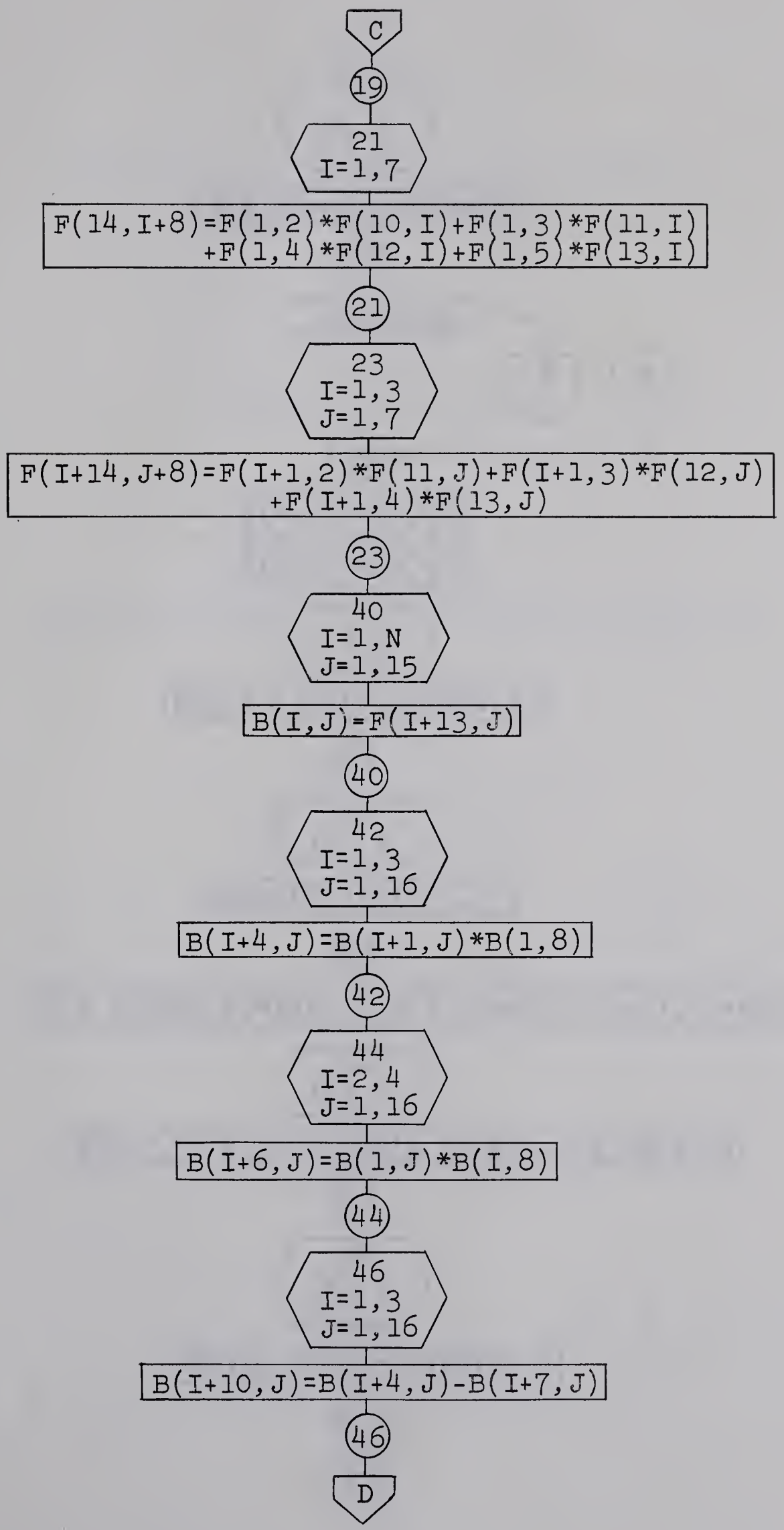
A.1 MAIN PROGRAM

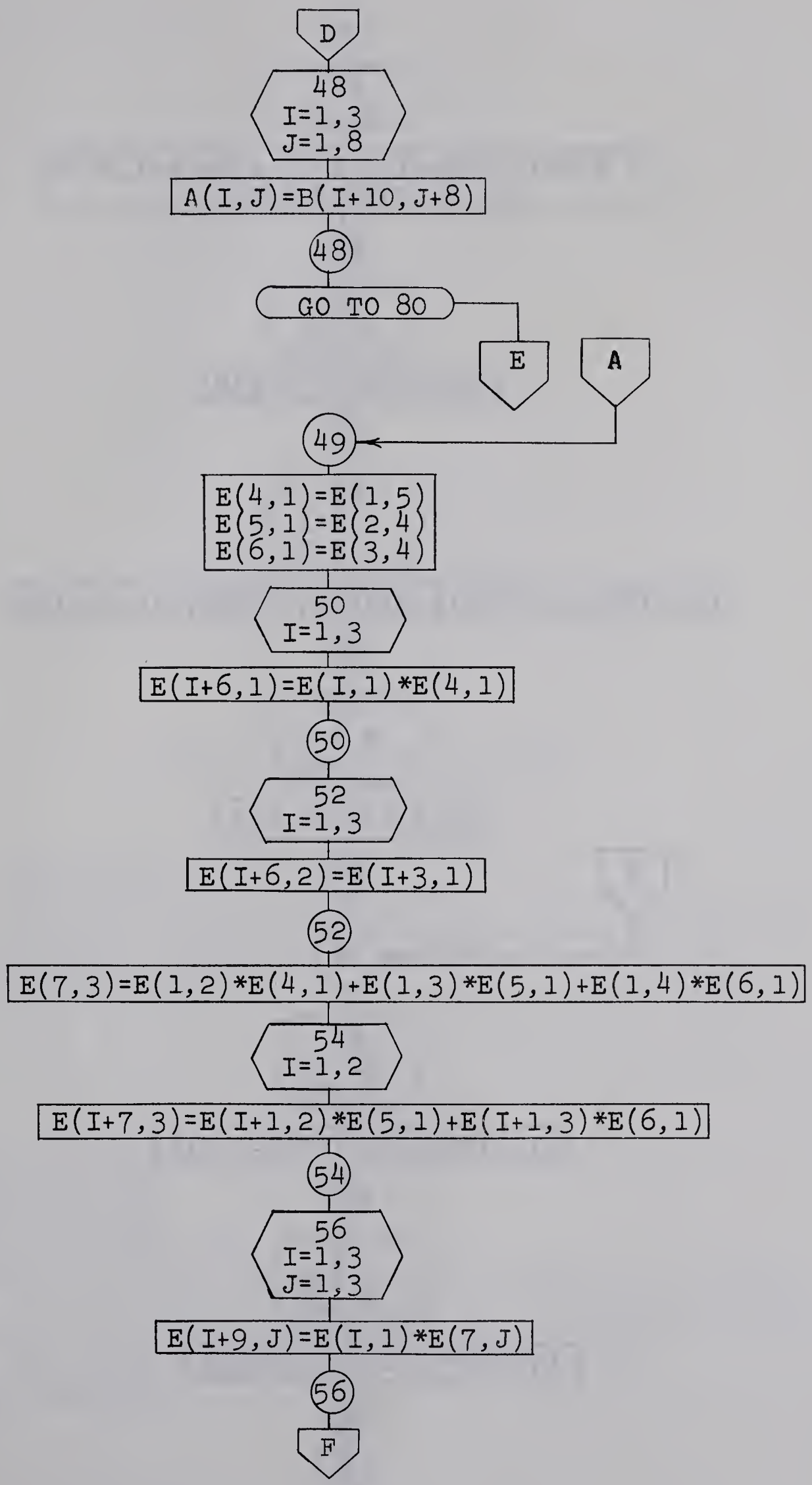


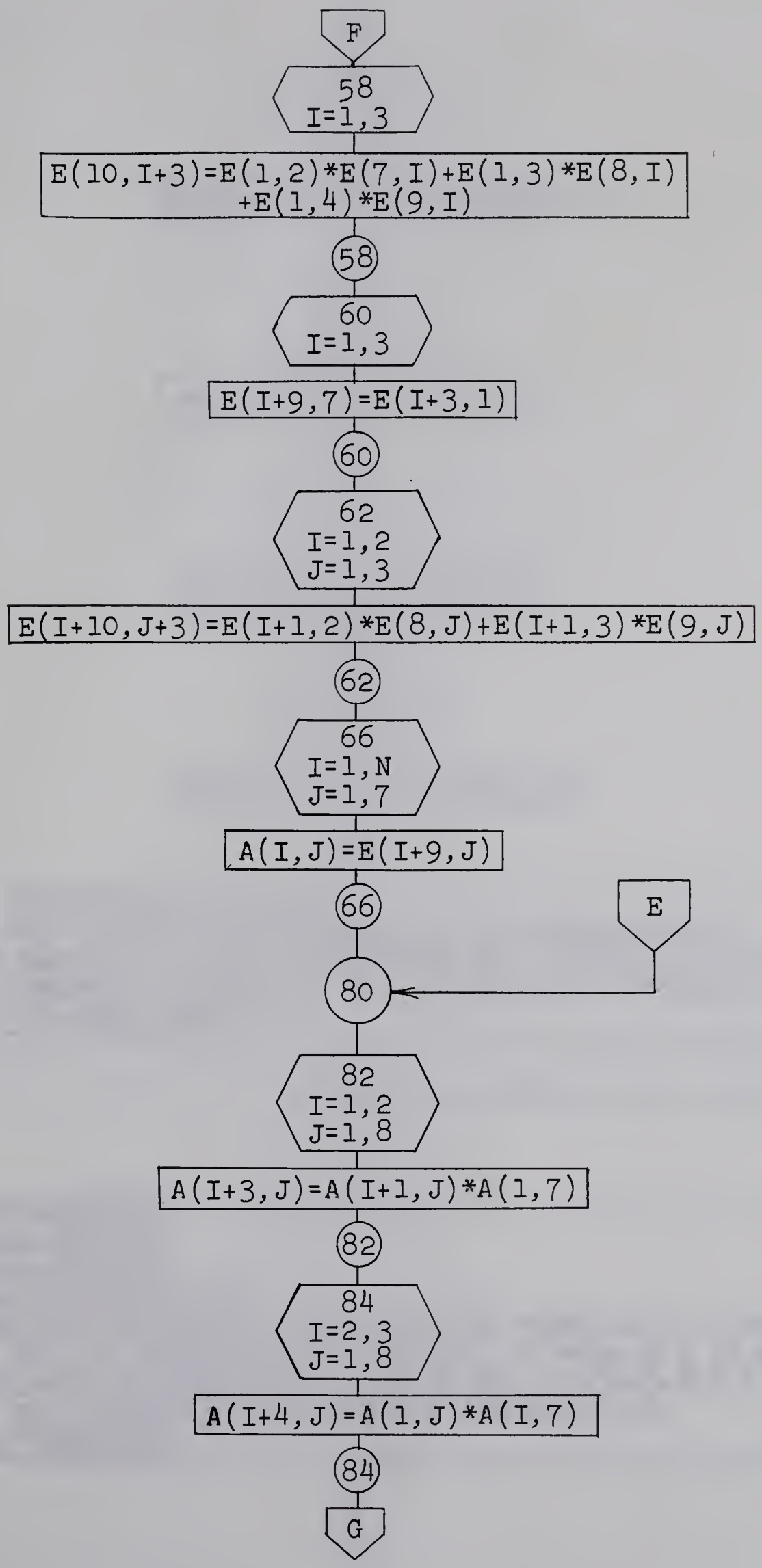


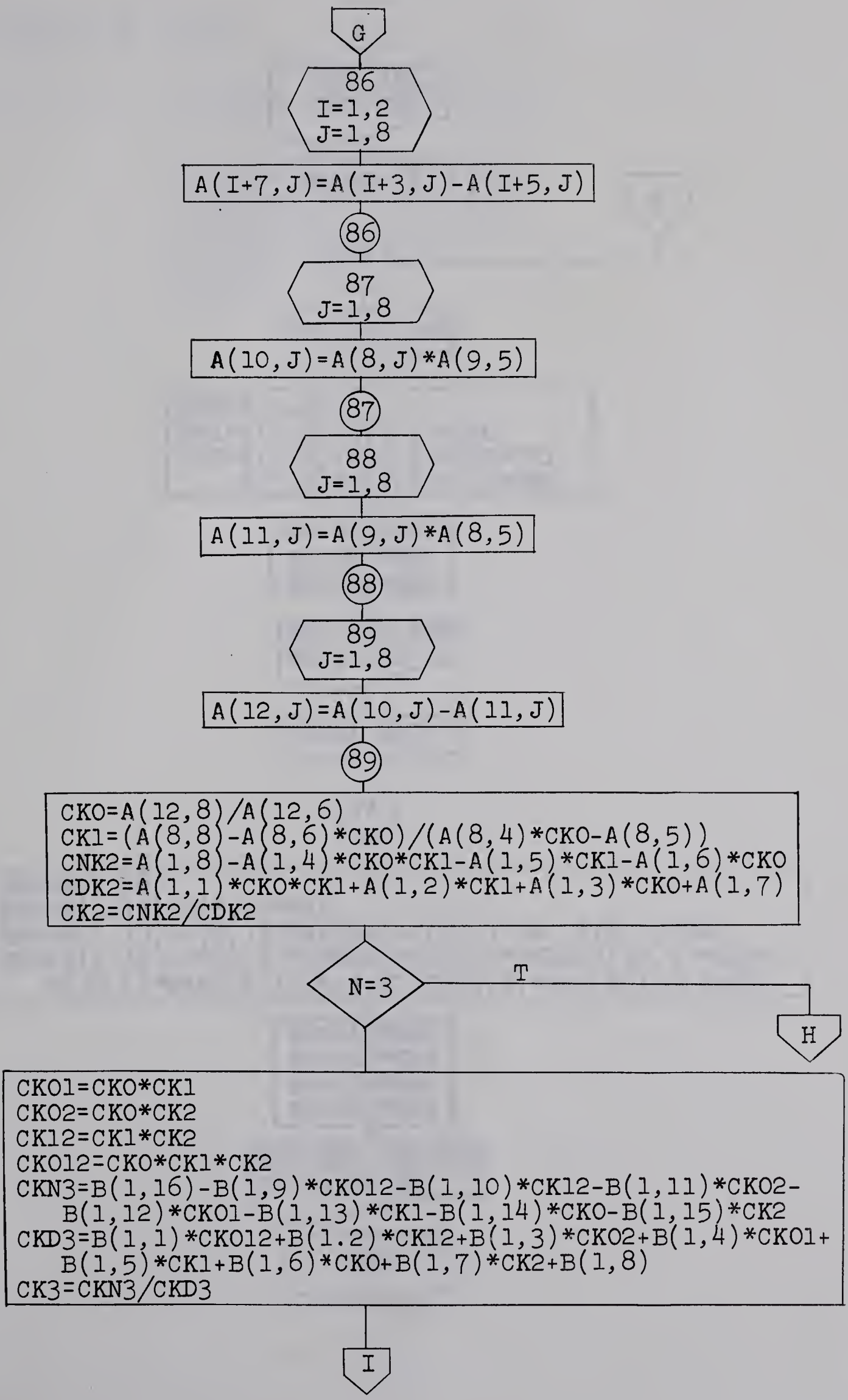

B
F(I+5,3)=F(I,2)*F(5,2)+F(I,3)*F(5,3)
+F(I,4)*F(5,4)
7
9
I=1,N
J=1,3
F(I+9,J)=F(I,1)*F(6,J)
9
11
I=1,3
F(10,I+3)=F(1,2)*F(6,I)+F(1,3)*F(7,I)
+F(1,4)*F(8,I)+F(1,5)*F(9,I)
11
F(10,7)=F(1,6)
13
I=1,3
J=1,3
F(I+10,J+3)=F(I+1,2)*F(7,J)+F(I+1,3)*F(8,J)
+F(I+1,4)*F(9,J)
13
15
I=1,3
F(I+10,7)=F(I+1,5)
15
17
I=1,4
J=1,7
F(I+13,J)=F(I,1)*F(10,J)
17
19
I=1,4
F(I+13,8)=F(I+9,7)
C

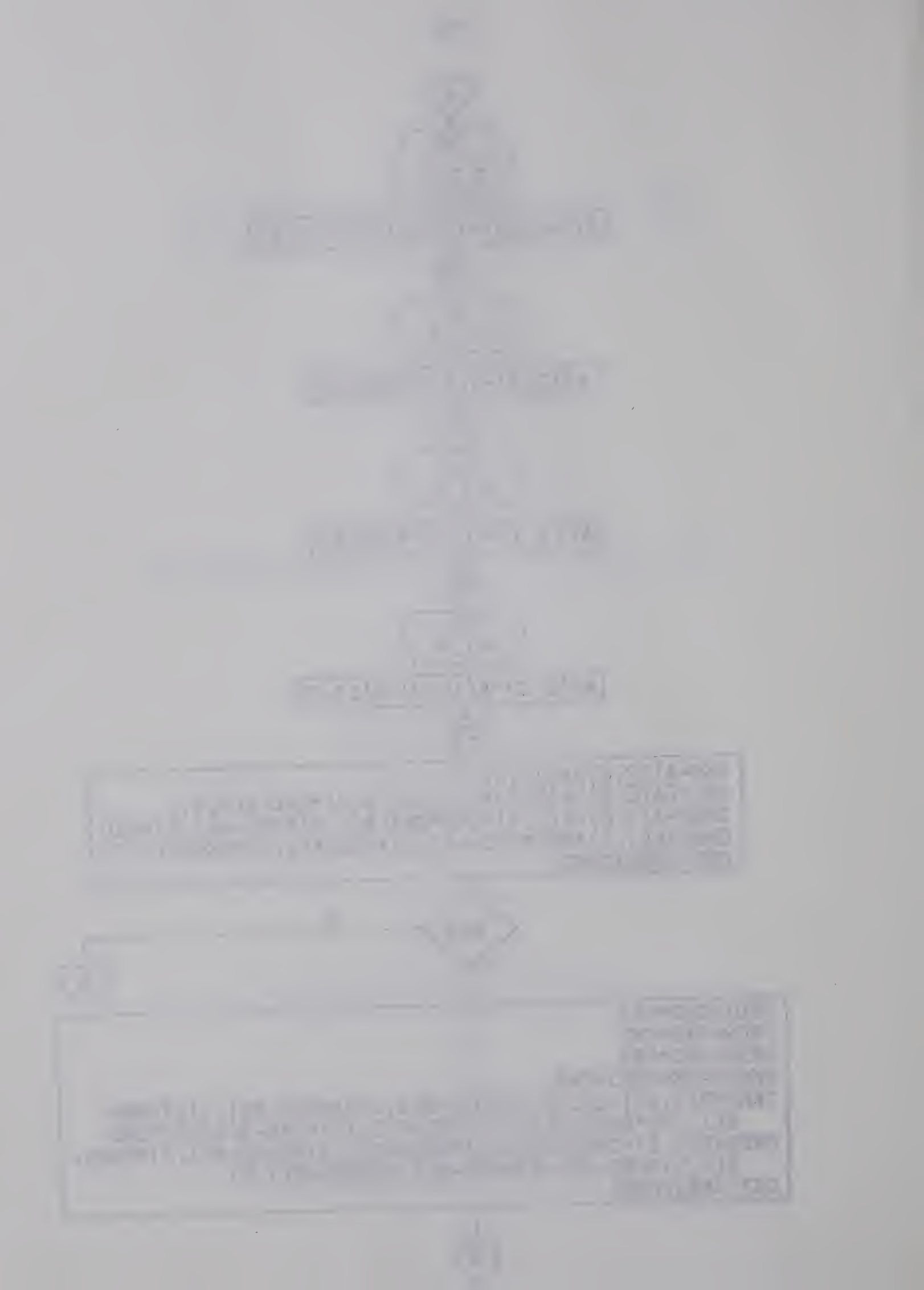


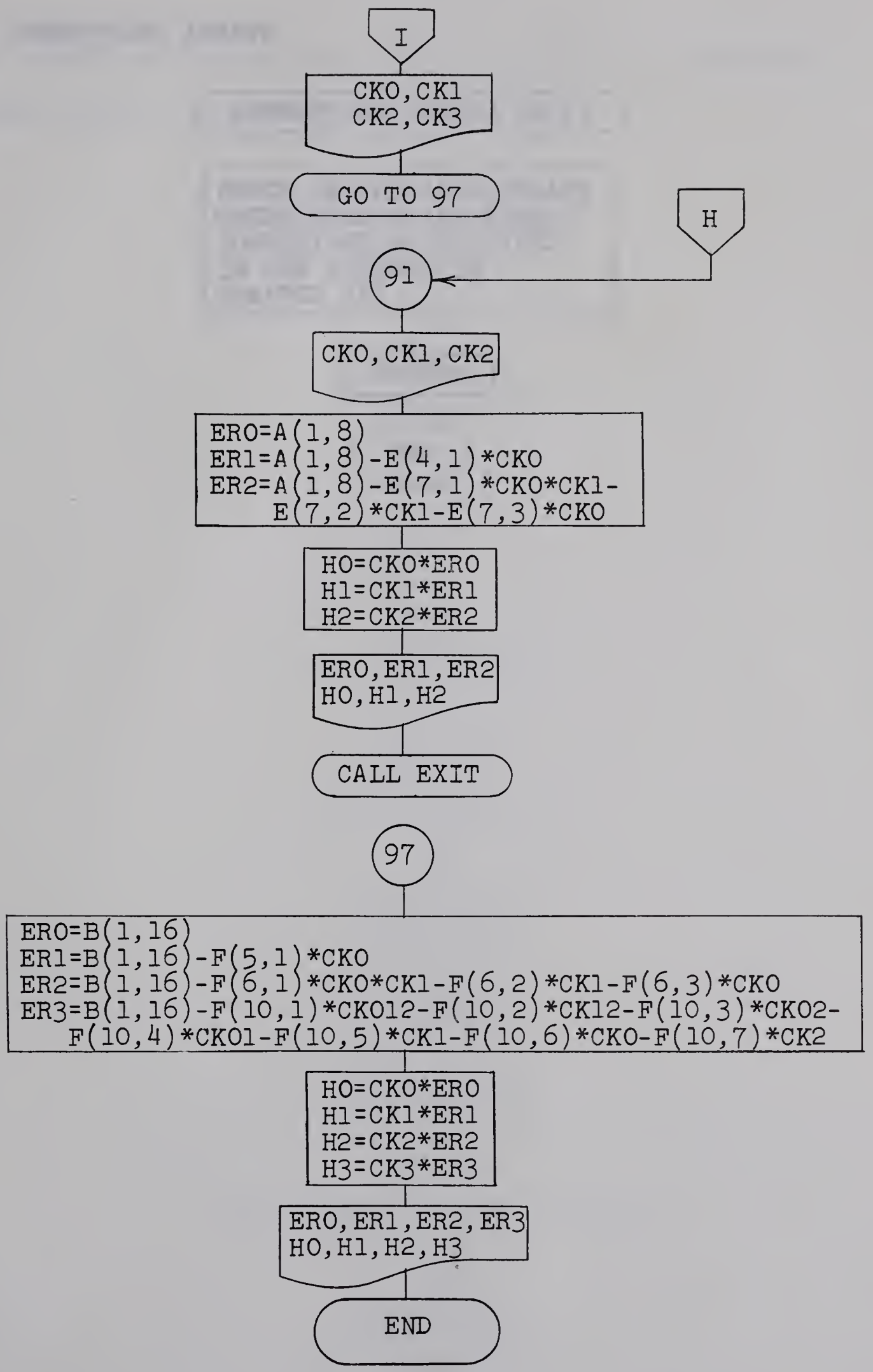

C
19
21
I=1,7
F(14,I+8)=F(1,2)*F(10,I)+F(1,3)*F(11,I)
+F(1,4)*F(12,I)+F(1,5)*F(13,I)
21
23
I=1,3
J=1,7
F(I+14,J+8)=F(I+1,2)*F(11,J)+F(I+1,3)*F(12,J)
+F(I+1,4)*F(13,J)
23
40
I=1,N
J=1,15
B(I,J)=F(I+13,J)
40
42
I=1,3
J=1,16
B(I+4,J)=B(I+1,J)*B(1,8)
42
44
I=2,4
J=1,16
B(I+6,J)=B(1,J)*B(I,8)
44
46
I=1,3
J=1,16
B(I+10,J)=B(I+4,J)-B(I+7,J)
46
D


D
48
I=1,3
J=1,8
A(I,J)=B(I+10,J+8)
48
GO TO 80
E
A
49
E(4,1)=E(1,5)
E(5,1)=E(2,4)
E(6,1)=E(3,4)
50
I=1,3
E(I+6,1)=E(I,1)*E(4,1)
50
52
I=1,3
E(I+6,2)=E(I+3,1)
52
E(7,3)=E(1,2)*E(4,1)+E(1,3)*E(5,1)+E(1,4)*E(6,1)
54
I=1,2
E(I+7,3)=E(I+1,2)*E(5,1)+E(I+1,3)*E(6,1)
54
56
I=1,3
J=1,3
E(I+9,J)=E(I,1)*E(7,J)
56
F

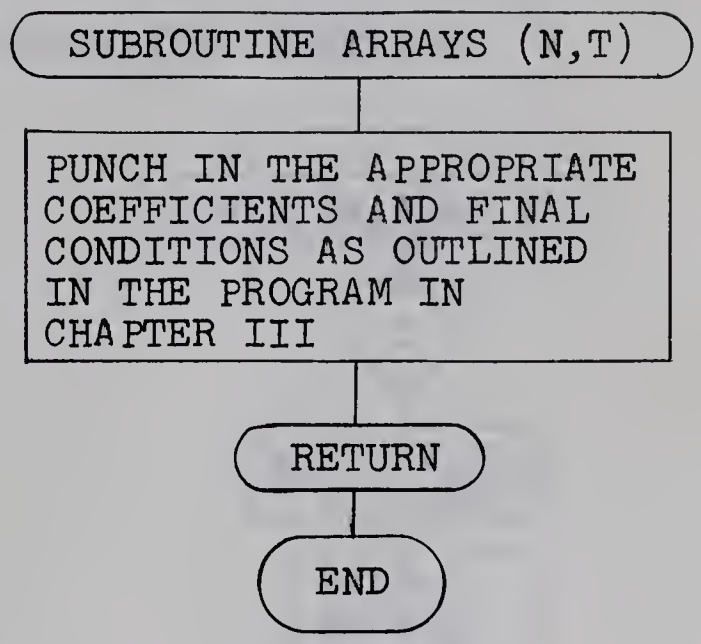

F
58
I=1,3
E(10,I+3)=E(1,2)*E(7,I)+E(1,3)*E(8,I)
+E(1,4)*E(9,I)
58
60
I=1,3
E(I+9,7)=E(I+3,1)
60
62
I=1,2
J=1,3
E(I+10,J+3)=E(I+1,2)*E(8,J)+E(I+1,3)*E(9,J)
62
66
I=1,N
J=1,7
A(I,J)=E(I+9,J)
66
E
80
82
I=1,2
J=1,8
A(I+3,J)=A(I+1,J)*A(1,7)
82
84
I=2,3
J=1,8
A(I+4,J)=A(1,J)*A(I,7)
84
G


G
86
I=1,2
J=1,8
A(I+7,J)=A(I+3,J)-A(I+5,J)
86
87
J=1,8
A(10,J)=A(8,J)*A(9,5)
87
88
J=1,8
A(11,J)=A(9,J)*A(8,5)
88
89
J=1,8
A(12,J)=A(10,J)-A(11,J)
89
CKO=A(12,8)/A(12,6)
CK1=(A(8,8)-A(8,6)*CKO)/(A(8,4)*CKO-A(8,5))
CNK2=A(1,8)-A(1,4)*CKO*CK1-A(1,5)*CK1-A(1,6)*CKO
CDK2=A(1,1)*CKO*CK1+A(1,2)*CK1+A(1,3)*CKO+A(1,7)
CK2=CNK2/CDK2
N=3
T
H
CKO1=CKO*CK1
CKO2=CKO*CK2
CK12=CK1*CK2
CKO12=CKO*CK1*CK2
CKN3=B(1,16)-B(1,9)*CKO12-B(1,10)*CK12-B(1,11)*CKO2-
B(1,12)*CKO1-B(1,13)*CK1-B(1,14)*CKO-B(1,15)*CK2
CKD3=B(1,1)*CKO12+B(1.2)*CK12+B(1,3)*CKO2+B(1,4)*CKO1+
B(1,5)*CK1+B(1,6)*CKO+B(1,7)*CK2+B(1,8)
CK3=CKN3/CKD3
I



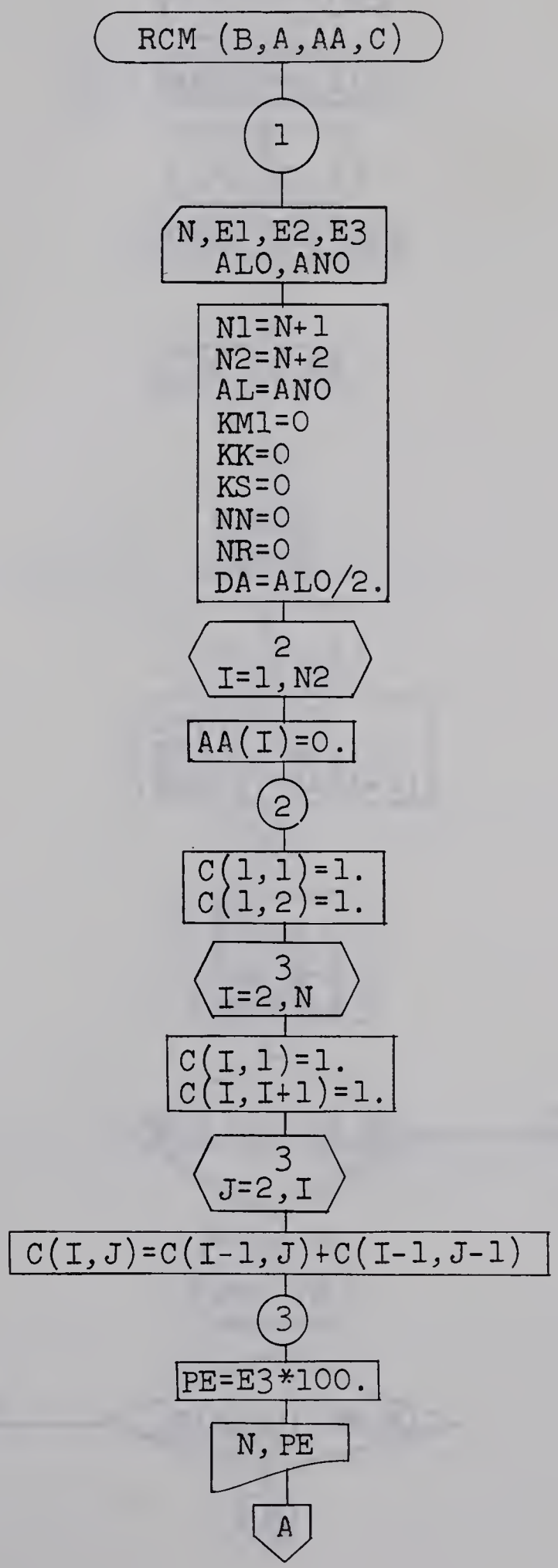

I
CKO,CK1
CK2,CK3
GO TO 97
H
91
CKO,CK1,CK2
ERO=A(1,8)
ER1=A(1,8)-E(4,1)*CKO
ER2=A(1,8)-E(7,1)*CKO*CK1-
E(7,2)*CK1-E(7,3)*CKO
HO=CKO*ERO
H1=CK1*ER1
H2=CK2*ER2
ERO,ER1,ER2
HO,H1,H2
CALL EXIT
97
ERO=B(1,16)
ER1=B(1,16)-F(5,1)*CKO
ER2=B(1,16)-F(6,1)*CKO*CK1-F(6,2)*CK1-F(6,3)*CKO
ER3=B(1,16)-F(10,1)*CKO12-F(10,2)*CK12-F(10,3)*CKO2-
F(10,4)*CKO1-F(10,5)*CK1-F(10,6)*CKO-F(10,7)*CK2
HO=CKO*ERO
H1=CK1*ER1
H2=CK2*ER2
H3=CK3*ER3
ERO,ER1,ER2,ER3
HO,H1,H2,H3
END

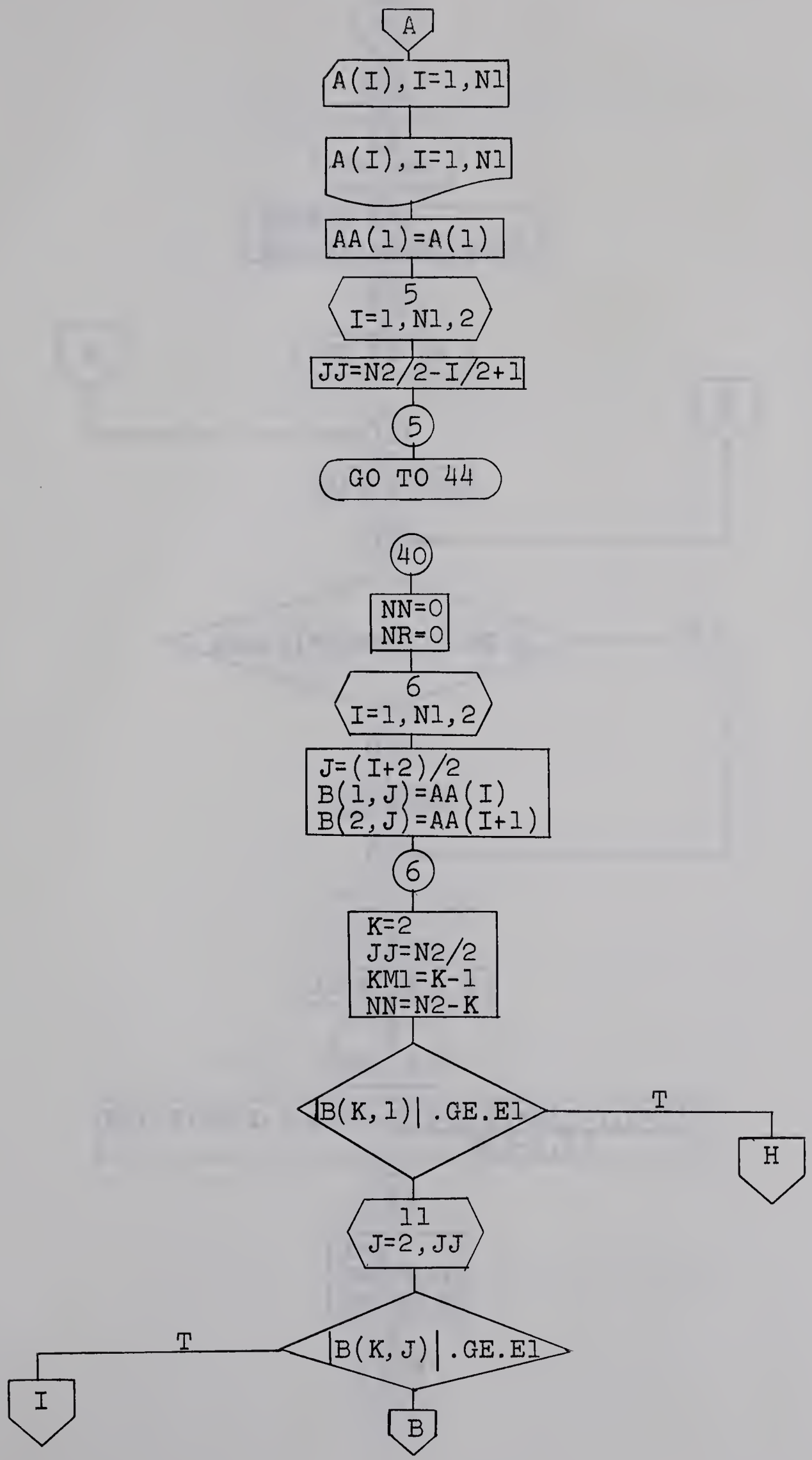
A.2 SUBROUTINE ARRAYS

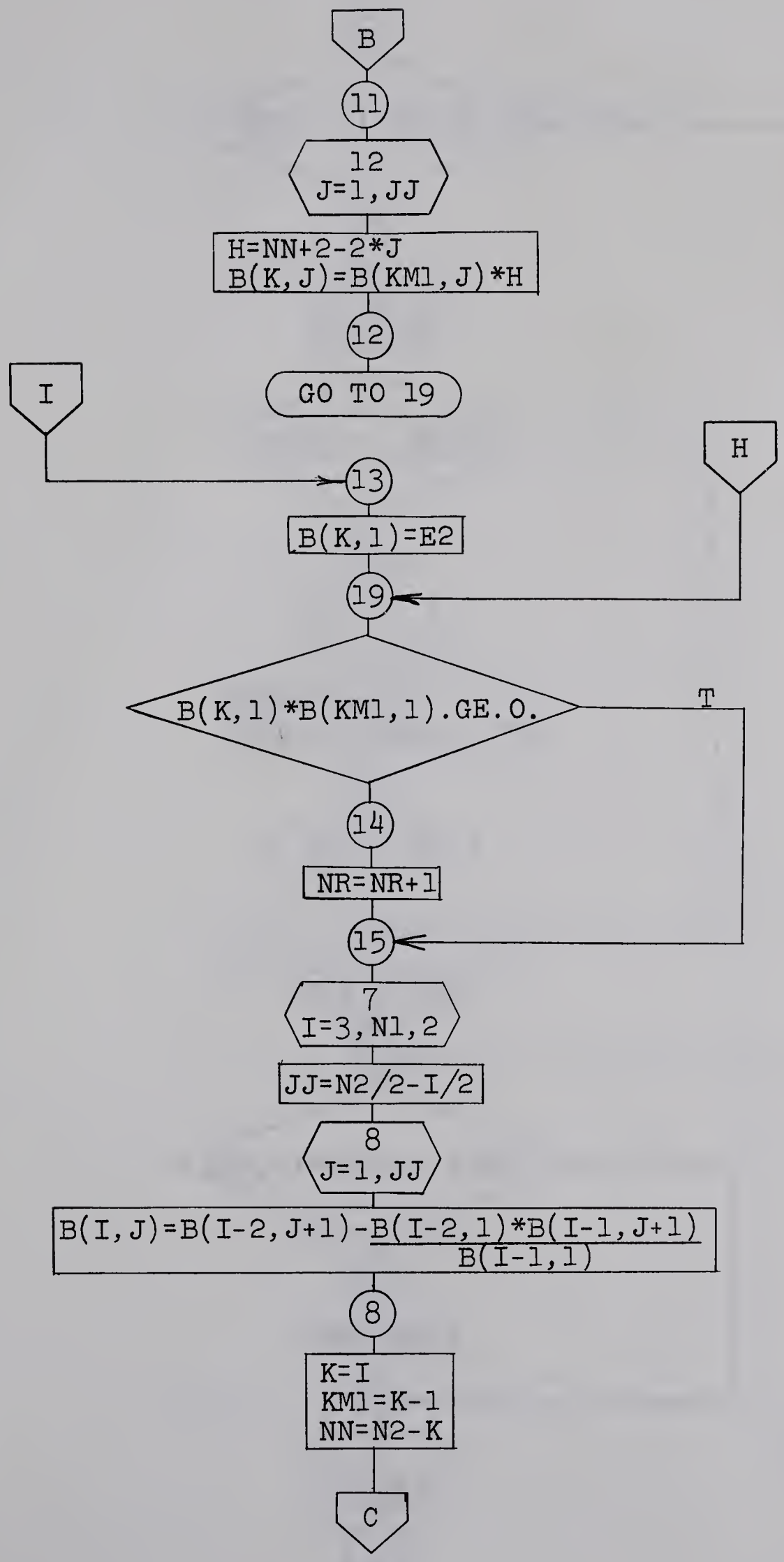


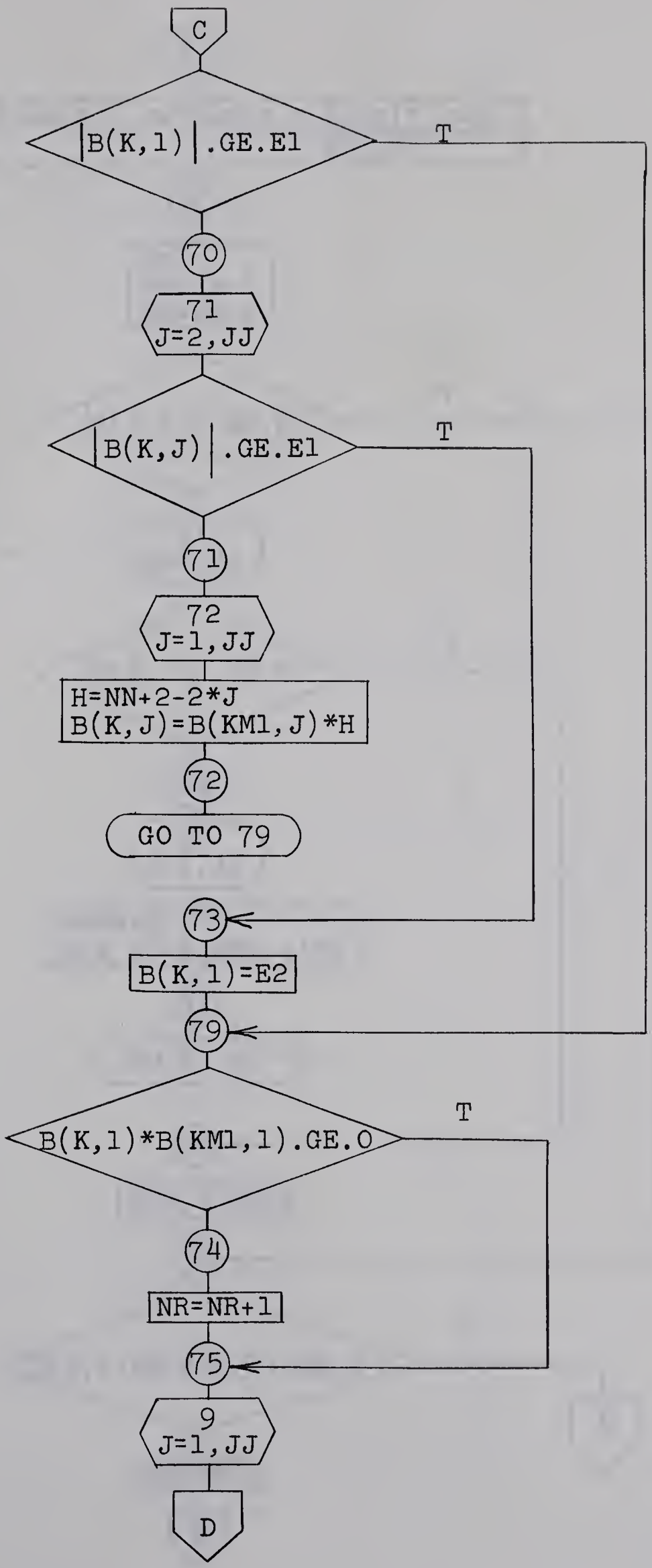
APPENDIX B

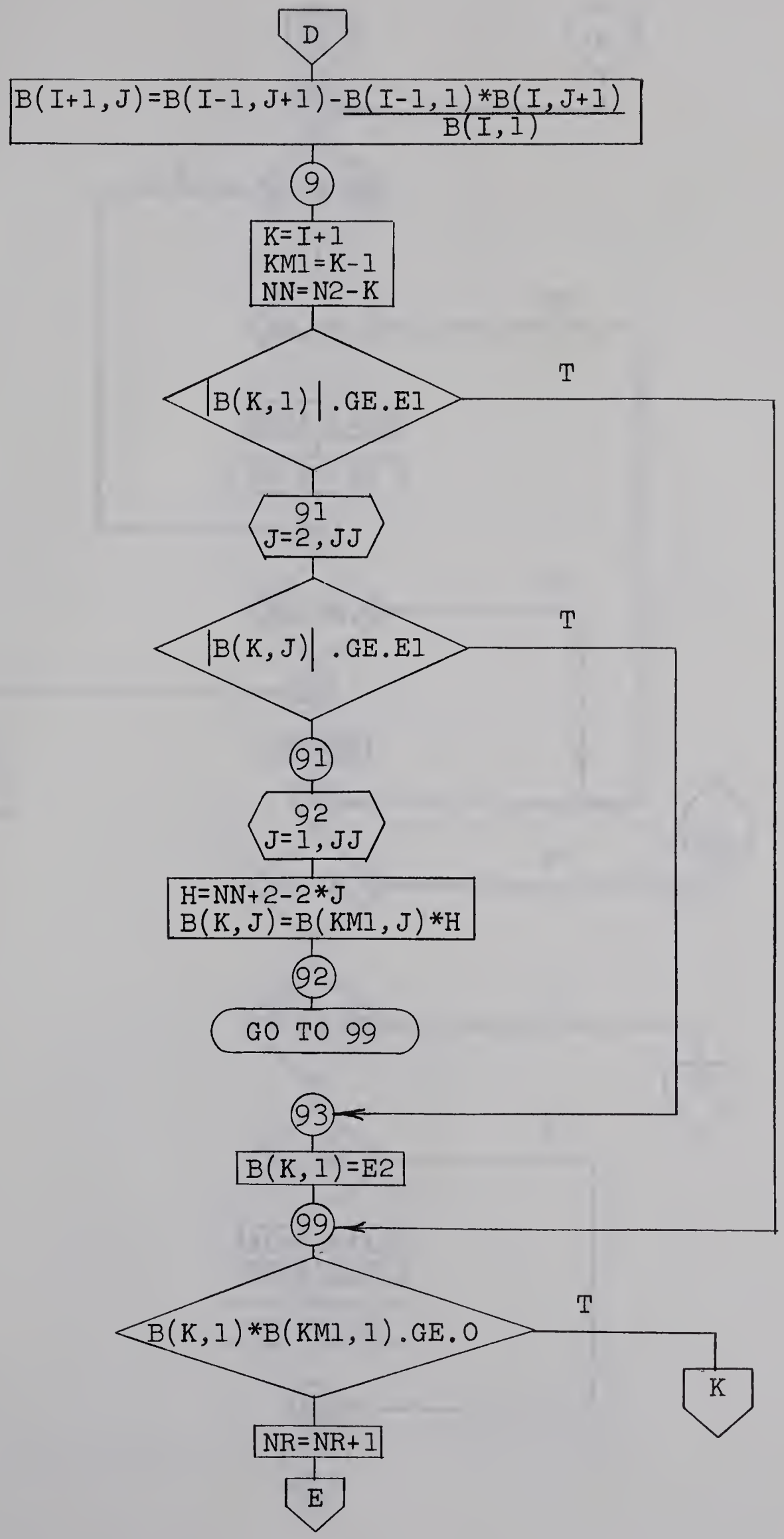
FLOW CHART FOR PROGRAM TO FIND THE ROOTS OF A POLYNOMIAL USING ROUTH'S STABILITY CRITERION

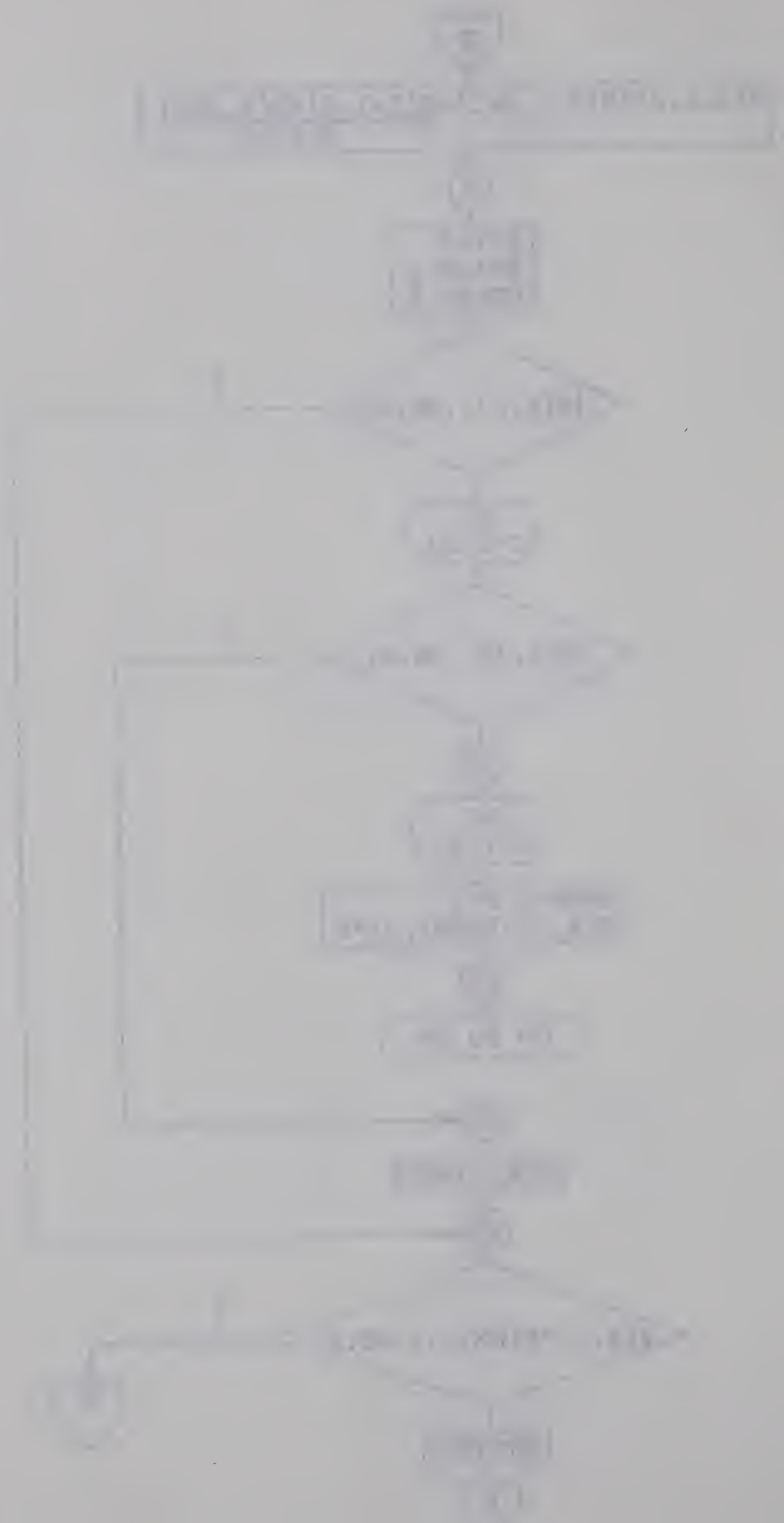


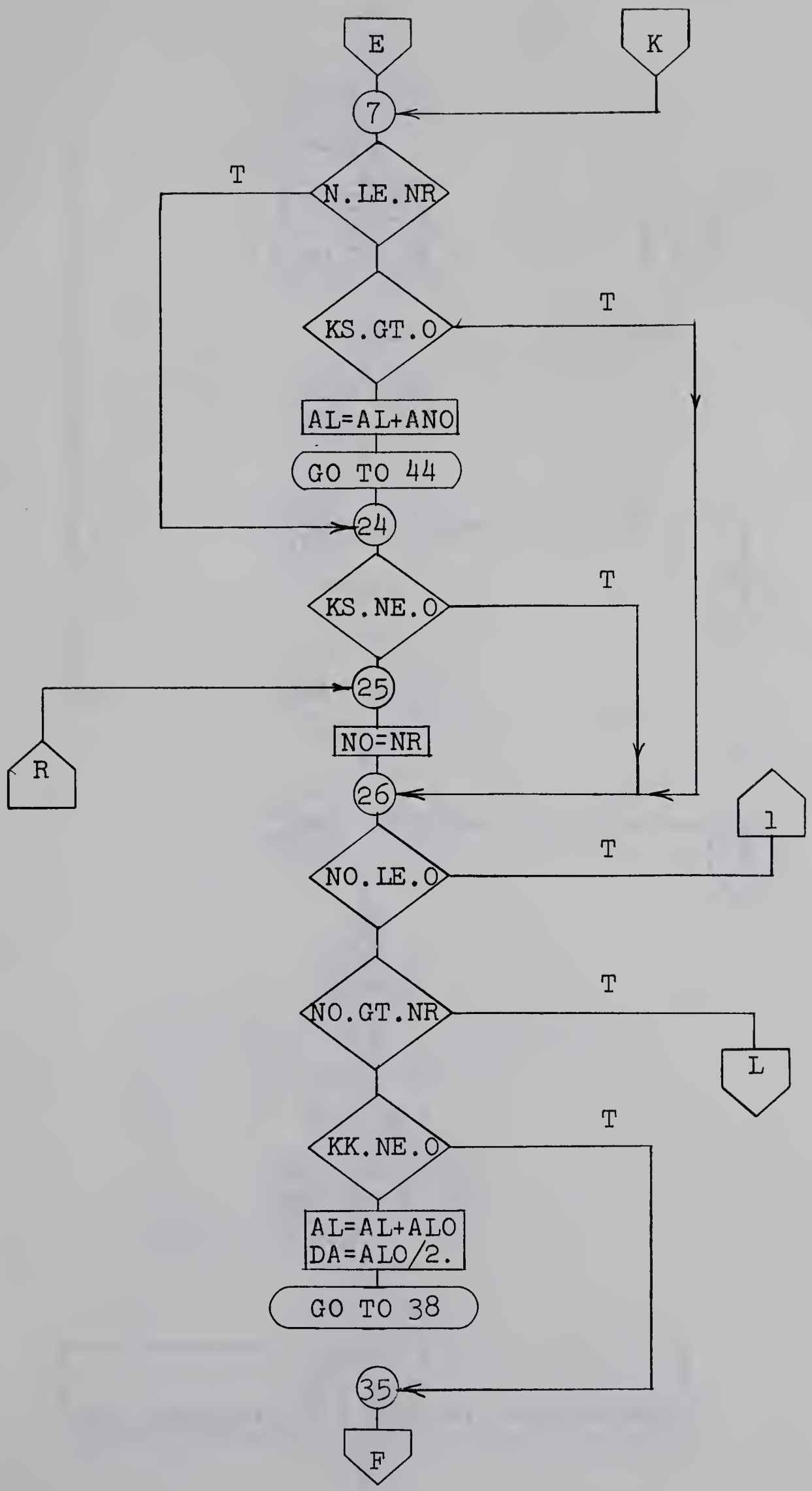

A
A(I), I=1, N1
A(I), I=1, N1
AA(1)=A(1)
5
I=1, N1, 2
JJ=N2/2-I/2+1
5
GO TO 44
40
NN=0
NR=0
6
I=1, N1, 2
J=(I+2)/2
B(1, J)=AA(I)
B(2, J)=AA(I+1)
6
K=2
JJ=N2/2
KM1=K-1
NN=N2-K
|B(K, 1)| .GE. E1
T
H
11
J=2, JJ
|B(K, J)| .GE. E1
T
I
B

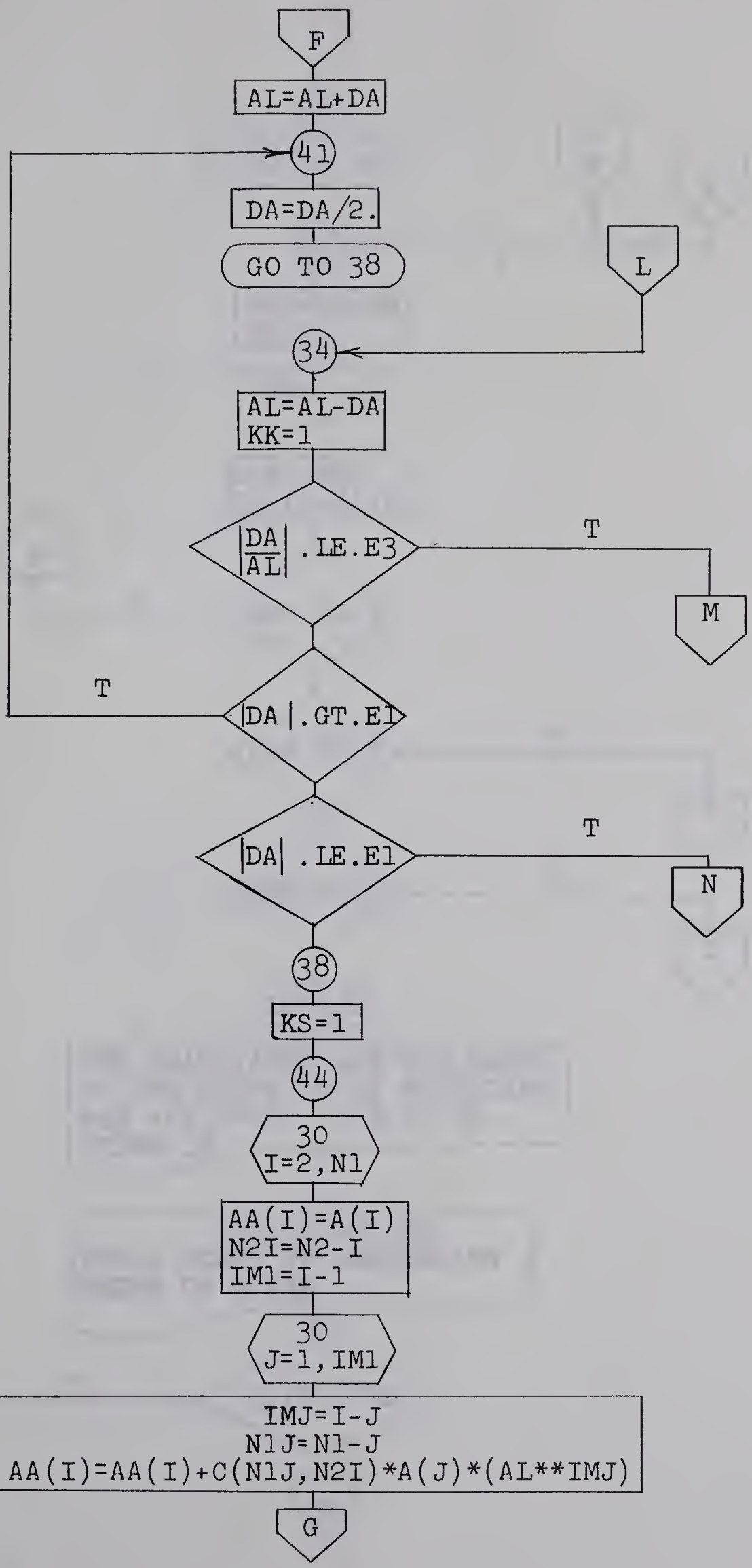

B
11
12
J=1,JJ
H=NN+2-2*J
B(K,J)=B(KM1,J)*H
12
GO TO 19
I
H
13
B(K,1)=E2
19
B(K,1)*B(KM1,1).GE.0.
T
14
NR=NR+1
15
7
I=3,N1,2
JJ=N2/2-I/2
8
J=1,JJ
B(I,J)=B(I-2,J+1)-B(I-2,1)*B(I-1,J+1)/B(I-1,1)
8
K=I
KM1=K-1
NN=N2-K
C

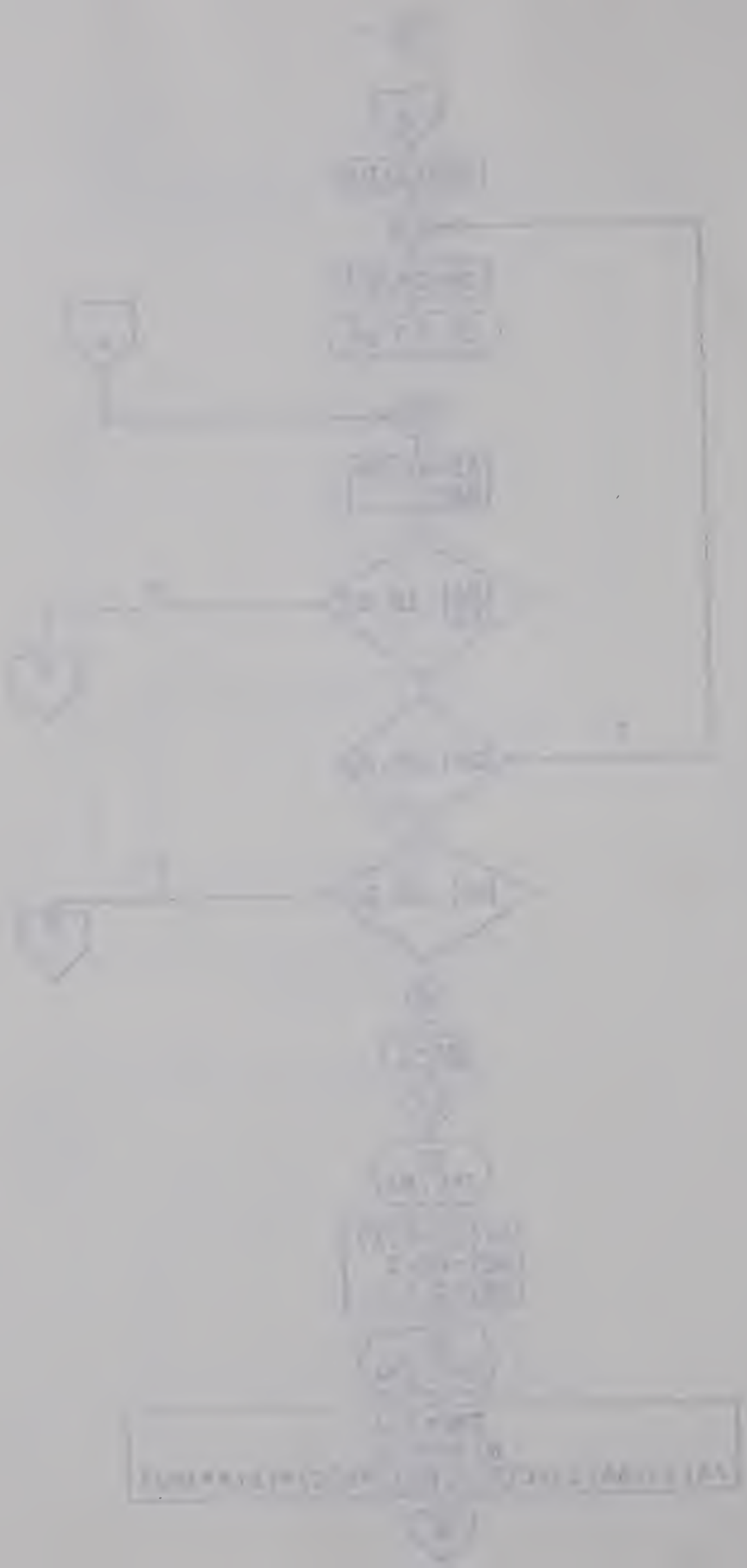

C
|B(K,1)|.GE.E1
T
70
71
J=2,JJ
|B(K,J)|.GE.E1
T
71
72
J=1,JJ
H=NN+2-2*J
B(K,J)=B(KM1,J)*H
72
GO TO 79
73
B(K,1)=E2
79
B(K,1)*B(KM1,1).GE.0
T
74
NR=NR+1
75
9
J=1,JJ
D

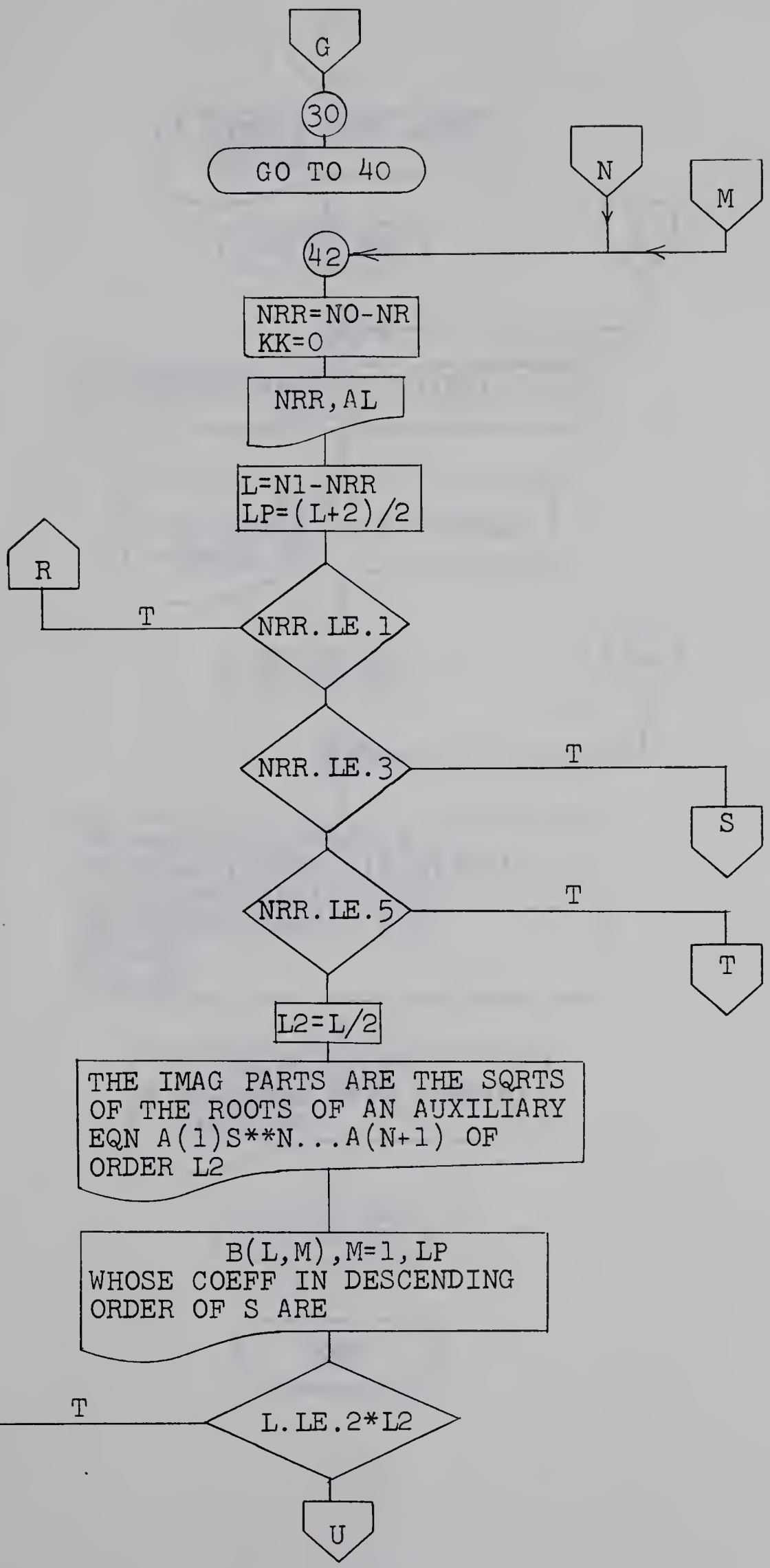

D
B(I+1,J)=B(I-1,J+1)-B(I-1,1)*B(I,J+1)/B(I,1)
9
K=I+1
KM1=K-1
NN=N2-K
|B(K,1)| .GE.E1
T
91
J=2,JJ
|B(K,J)| .GE.E1
T
91
92
J=1,JJ
H=NN+2-2*J
B(K,J)=B(KM1,J)*H
92
GO TO 99
93
B(K,1)=E2
99
B(K,1)*B(KM1,1).GE.0
T
K
NR=NR+1
E

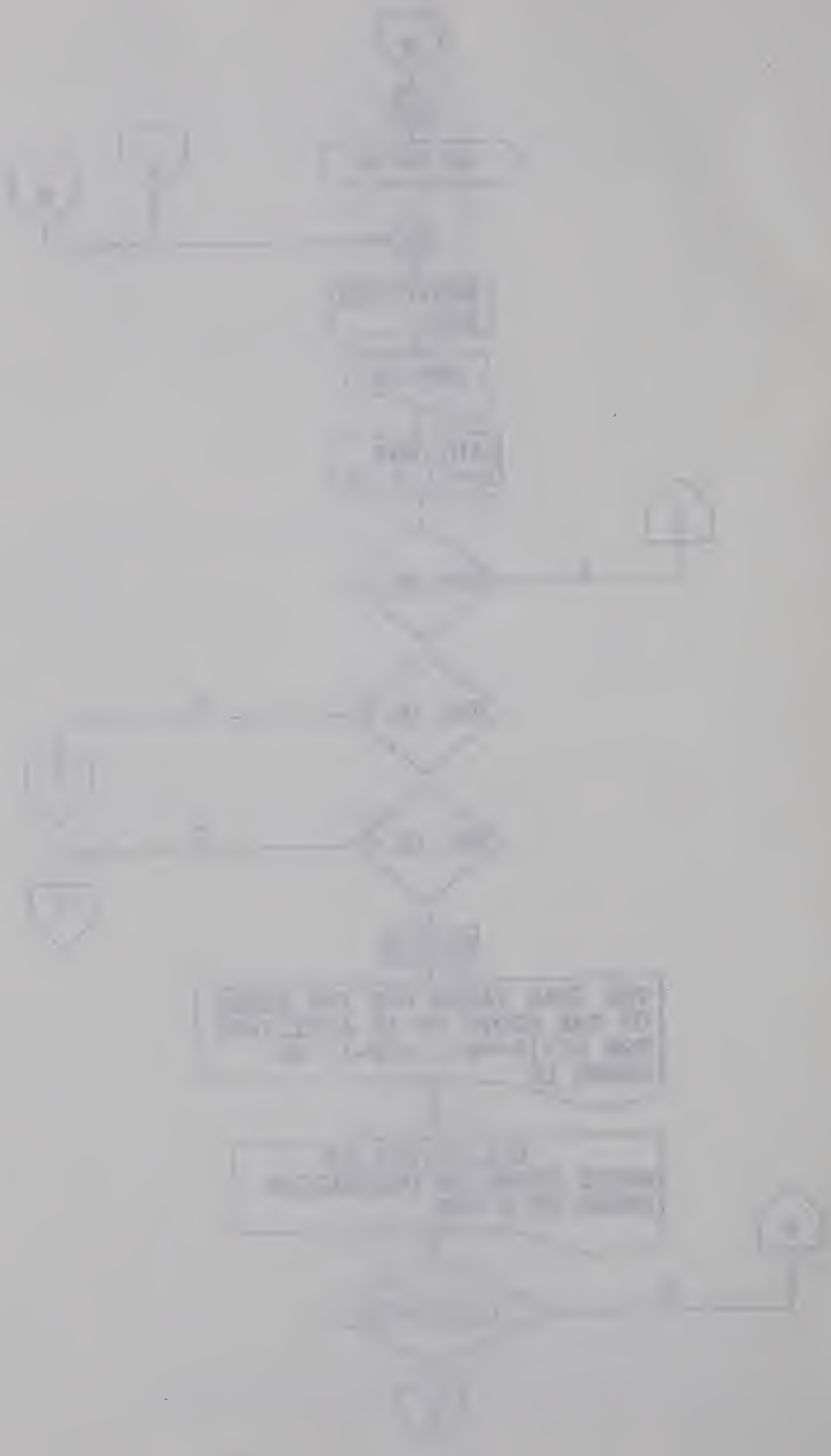


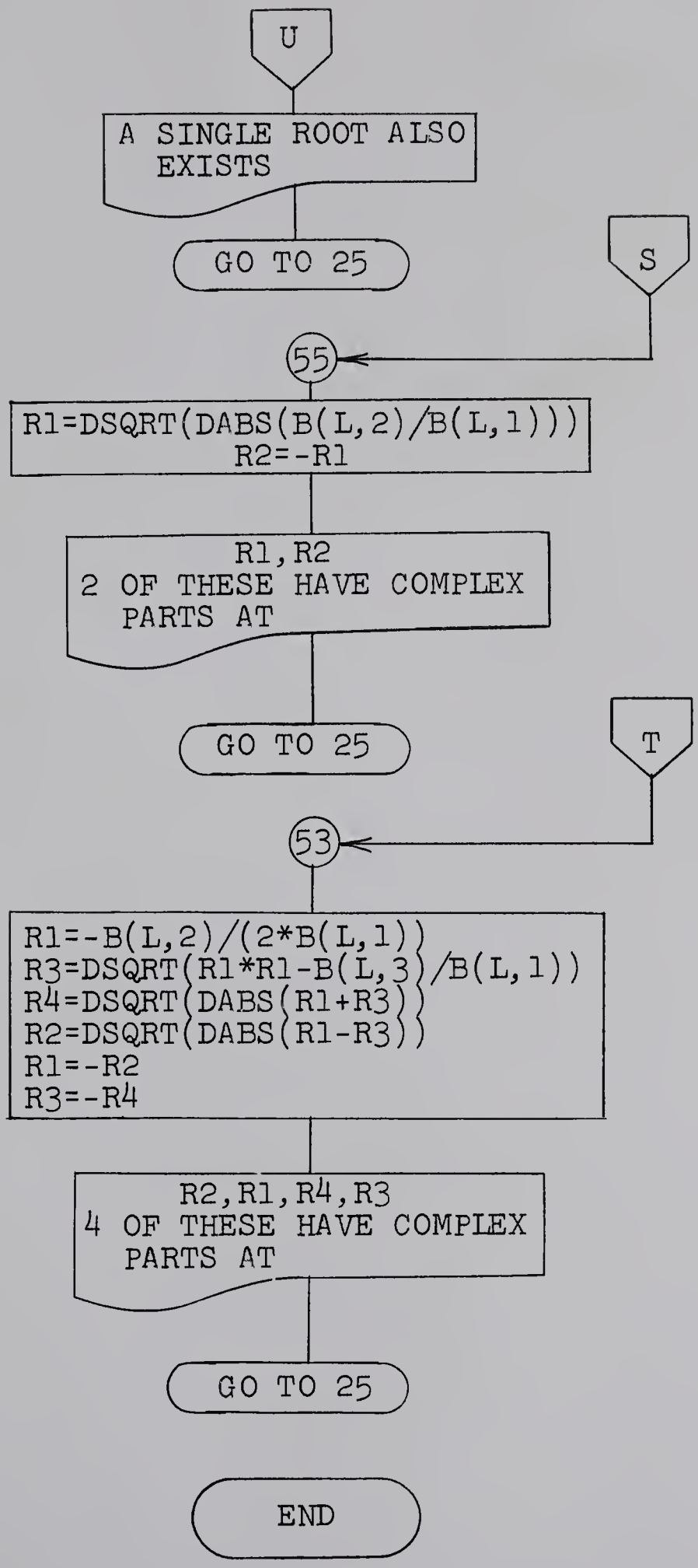

E
K
7
T
N.LE.NR
T
KS.GT.O
AL=AL+ANO
GO TO 44
24
T
KS.NE.O
25
R
NO=NR
26
1
T
NO.LE.O
T
NO.GT.NR
L
T
KK.NE.O
AL=AL+ALO
DA=ALO/2.
GO TO 38
35
F


F
AL=AL+DA
41
DA=DA/2.
GO TO 38
L
34
AL=AL-DA
KK=1
|DA/AL| .LE.E3
T
M
T
|DA| .GT.E1
|DA| .LE.E1
T
N
38
KS=1
44
30
I=2,N1
AA(I)=A(I)
N2I=N2-I
IM1=I-1
30
J=1,IM1
IMJ=I-J
N1J=N1-J
AA(I)=AA(I)+C(N1J,N2I)*A(J)*(AL**IMJ)
G




G
30
GO TO 40
N
M
42
NRR=NO-NR
KK=0
NRR,AL
L=N1-NRR
LP=(L+2)/2
R
T
NRR.LE.1
T
NRR.LE.3
S
T
NRR.LE.5
T
L2=L/2
THE IMAG PARTS ARE THE SQRTS
OF THE ROOTS OF AN AUXILIARY
EQN A(1)S**N...A(N+1) OF
ORDER L2
B(L,M),M=1,LP
WHOSE COEFF IN DESCENDING
ORDER OF S ARE
R
T
L.LE.2*L2
U




U
A SINGLE ROOT ALSO EXISTS
GO TO 25
S
55
R1=DSQRT(DABS(B(L,2)/B(L,1)))
R2=-R1
R1,R2
2 OF THESE HAVE COMPLEX PARTS AT
GO TO 25
T
53
R1=-B(L,2)/(2*B(L,1))
R3=DSQRT(R1*R1-B(L,3)/B(L,1))
R4=DSQRT(DABS(R1+R3))
R2=DSQRT(DABS(R1-R3))
R1=-R2
R3=-R4
R2,R1,R4,R3
4 OF THESE HAVE COMPLEX PARTS AT
GO TO 25
END